VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

APLIKACE PRO VEDENÍ ČTENÁŘSKÉHO DENÍKU

A READING DIARY APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PETR NAVRÁTIL**
AUTHOR

VEDOUCÍ PRÁCE — **MARTIN KRČMA, Ing.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19681/2016/xnavra53

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Navrátil Petr**
Obor: Informační technologie
Téma: **Aplikace pro vedení čtenářského deníku**
A Reading Diary Application
Kategorie: Softwarové inženýrství

Pokyny:

1. Nastudujte problematiku vedení čtenářského deníku a dostupná řešení.
2. Navrhněte multiplatformní aplikaci pro vedení čtenářského deníku, která umožní vedení záznamu o přečtených knihách, rozečtených knihách a knihách k přečtení. Bude podporovat řazení a výpisy dle různých kategorií. Aplikace bude k zadaným knihám automaticky hledat informace online a každé knize tak přiřadí obrázek obalu, počet stran, možnosti nákupu apod. Dále bude generovat statistiky nejen o rychlosti čtení. Aplikace bude vybavena moderním vícejazyčným grafickým uživatelským rozhraním. Dále umožní export seznamů knih do textového souboru a do pdf.
3. Aplikaci implementujte.
4. Proveďte uživatelské testování aplikace a vyhodnoťte výsledky.
5. Navrhněte další možné směřování práce.

Literatura:

- dle pokynů vedoucího

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Krčma Martin, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Díky rozvoji informačních technologií je možné přesouvat dříve ryze psané poznámky do elektronické formy. Jedním z příkladů může být čtenářský deník, se kterým většina z nás přišla do styku na základní škole právě v psané formě. Tato práce se snaží o vytvoření aplikace vhodné k vedení čtenářského deníku v elektronické formě s možností přidávání vlastních literárních rozborů a možností sledovat dobu čtení jednotlivých knih. Knihy je navíc možné přidávat do vlastních seznamů—poliček, což umožňuje vytvoření malé elektronické knihovny, kterou je možné sdílet mezi ostatními uživateli.

Abstract

Development of information technology enables us to take notes of all kind in more comfortable, electronic way than as it was in written form. Improvement can be seen for example in case of reading diaries which were mostly hand-written when we first got in touch with them in elementary school. This Bachelor's thesis is focusing on creating an application suitable for writing a personal reading diary. Application will enable user to fill own literary analysis, track reading time of books or add books to user's individual lists which will represent shelves as known from libraries. Users will be able to share their books and shelves with other users.

Klíčová slova

čtenářský deník, kniha, polička, stopování času, Goodreads, Google Books, Databazeknih.cz, Angular 2, Go, Auth0, JWT, PDF

Keywords

reading diary, book, shelf, time tracking, Goodreads, Google Books, Databazeknih.cz, Angular 2, Go, Auth0, JWT, PDF

Citace

NAVRÁTIL, Petr. *Aplikace pro vedení čtenářského deníku.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Krčma, Ing.

Aplikace pro vedení čtenářského deníku

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Krčmy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Navrátil
17. května 2017

Poděkování

Rád bych poděkoval panu Ing. Martinu Krčmovi za vedení této práce, věcné poznámky a rady, které mi poskytoval v průběhu vytváření práce. Poděkování patří i Tereze Vápeníkové za poskytnuté informace týkající se literárního rozboru.

Obsah

Kapitola 1

Úvod

S pojmem „čtenářský deník“ se člověk většinou setkává již na základní škole, kdy má vypracovávat krátká shrnutí o knihách, které přečetl za určité časové období. Vedení čtenářského deníku je v tu dobu koncipováno spíše jako hra, kdy žáci soutěží mezi sebou, což je vede k přečtení co nejvíce knih, čímž si pasivně zlepšují svoji čtenářskou gramotnost. Na středních školách přechází vedení čtenářský deníků z her na přípravu k blížícím se maturitním zkouškám. Deník nabývá pevné struktury, je kladen důraz na pochopení přečteného textu a schopnosti zařadit jej do správných literárních kategorií. Mezi tyto kategorie patří například *literární směr, žánr* či *druh*. Čtenářský deník ovšem nemusí být veden jen kvůli povinnostem pramenícím ze studia, ale i z osobních důvodu. Může sloužit například jako seznam přečtených knih a poznámek, které vyvstaly na mysl během čtení [16].

Vést si seznam knih v elektronické podobně není dnes již žádný problém. Elektronická forma navíc přináší i několik výhod oproti klasické psané formě. Deník může být rozšířen i o další seznamy knih, mezi kterými je možné knihy bezproblémově přesouvat, například seznam knih k budoucímu čtení.

Cílem této bakalářské práce je vytvořit aplikaci, která bude umožňovat uživateli vést si čtenářský deník v elektronické formě. Uživatel bude moci vyhledávat knihy, rozřazovat je do čtyř seznamů podle jejich stavu či je přidávat do vytvořených poliček pro větší přehlednost. Ke každému titulu bude moci uživatel přidat literární rozbor, který bude společně s informacemi o titulu možné exportovat ve formátu PDF. Exportovat bude možné i seznam knih. Čtení každé knihy navíc bude možné stopovat, čímž se uživateli naskytne možnost zobrazení doby čtení titulu na časové ose, případně detailní rozpis intervalů čtení. Pro sdílení knih ostatními uživateli bude aplikace podporovat systém kamarádů, který bude nabízet seznam knih a poliček vybraného kamaráda.

V kapitole č. 2 jsou popsány některé z existujících služeb, které se dají použít ke správě čtenářského deníku včetně chybějících požadavků vůči této aplikaci. Kapitola č. 3 se věnuje návrhu aplikace. Kapitola č. 4 navazuje vlastní implementací aplikace. Průběh a výsledky testování čtenářského deníku včetně návrhů na vylepšení jsou uvedeny v kapitole č. 5.

Kapitola 2

Konkurenční služby

Tato kapitola se zaměřuje na existující služby, které se týkají tématiky čtenářského deníku a zkoumá,. zda služba splňuje následující požadavky:

1. vyhledávat knihy,
2. přidávat knihy do seznamů podle jejich stavu,
3. přidávat knihy do poliček[1],
4. přidávat literární rozbor,
5. zobrazit průběh čtení knihy nebo dobu čtení knihy,
6. exportovat seznam knih.

Existující konkurenční služby se dají rozdělit do dvou kategorií podle nabízeného obsahu:

- **databáze knih** — služba, která poskytuje velké množství knih včetně informací o nich, knihy je zde možné přidávat do různých seznamů;
- **literární rozbor** — služba, kde jsou vypracované rozbory knih, ovšem knihy si nelze přidávat do seznamů či s nimi jakkoli pracovat.

Jelikož druhá kategorie splňuje pouze první a částečně čtvrtý požadavek na aplikaci, nebude již dále zkoumána. Nicméně do této kategorie spadá například portál **český-jazyk.cz**[2] nebo **seminarky.cz**[3].

Služby patřící do první kategorie většinou splňují všechny požadavky kromě možnosti přidat literární rozbor. Možnost sledování průběhu čtení knihy či doby čtení knihy bývá velice stroze řešena. V následujících podkapitolách budou představeny tři nejznámější služby patřící do této kategorie.

2.1 Goodreads

Goodreads je zahraniční služba, která vznikla v roce 2007 a patří společnosti **Amazon**[4]. Jedná se spíše o sociální síť zaměřenou na milovníky knih. Služba bohužel není dostupná

[1] Pojem polička v této práci zastupuje seznam knih, který má pro uživatele nějaký logický význam. Například se může jednat o seznam oblíbených knih nebo knih s určitou tématikou.

[2] http://www.cesky-jazyk.cz/ctenarsky-denik

[3] http://www.seminarky.cz/ctenarsky-denik

[4] https://www.amazon.com/

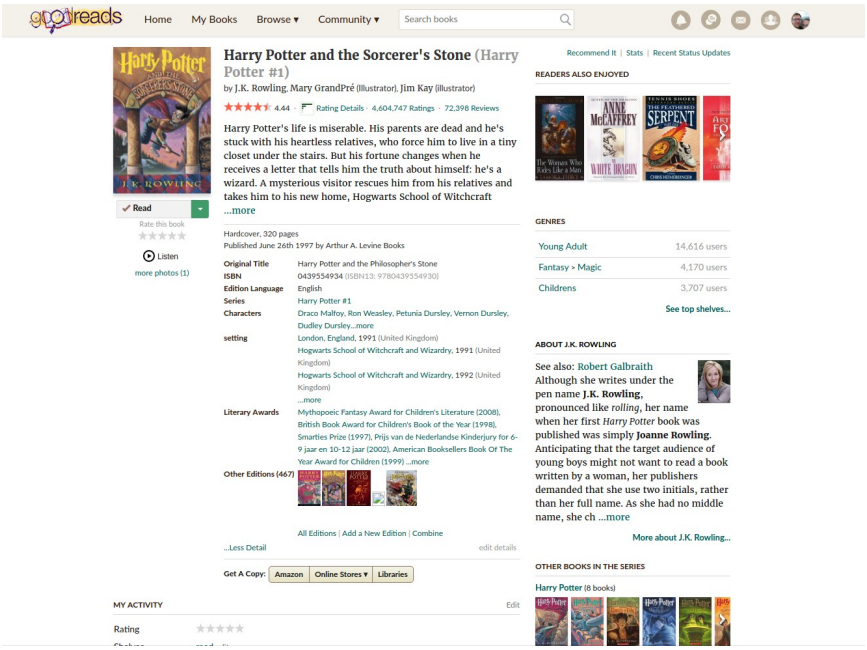
v českém jazyce, nicméně pokrytí českých titulů je obstojné. Pro plné využití služby je vyžadovaná registrace, která je bezplatná a přináší spoustu výhod [4].

Portál nabízí možnost vyhledávat knihy a následně je přidávat podle jejich stavu do jednotlivých seznamů, tedy vytvářet seznamy přečtených nebo čtených knih či knih, které by si uživatel rád přečetl v blízké budoucnosti. Dále umožňuje vytvářet si vlastní poličky, do kterých je knihy též možné zařazovat. Kromě knih také vede informace o autorech, což umožňuje prozkoumat tvorbu oblíbeného autora. Portál navíc sleduje, které knihy uživatel čte, a na základě toho doporučuje uživateli další knihy k přečtení.

Sekce detailu knihy nabízí informace o vydání vyhledaného titulu, stručné informace o autorovi a komentáře ostatních uživatelů, na které je možné reagovat. Nechybí doporučené podobné knihy, další knihy autora, případně knihy spadající do stejné série jako vyhledaná kniha.

Otázka sledování čtení knihy je řešena pouze částečně. Ve chvíli, kdy je kniha přidána do předdefinovaného seznamu aktuálně čtených knih, se zapamatuje doba tohoto přidání jako začátek čtení knihy a až v případě přesunutí této knihy do předdefinovaného seznamu přečtených knih je čtení knihy ukončeno. V detailu knihy je poté možné podívat se na tyto dvě data a případně je ručně změnit. Po celou dobu čtení je možné ručně upravovat na jaké stránce se uživatel aktuálně nachází.

Nechybí ani možnost interakce s ostatními uživateli. Uživatel může hodnotit a komentovat jednotlivé knihy, což může ostatní uživatele ovlivnit v rozhodování zda knihu číst či nikoli [10]. Goodreads dále nabízí možnost přidávání přátel, sledování jejich čtenářské aktivity či doporučení vašich oblíbených knih vašim přátelům.



Obrázek 2.1: Detail knihy Goodreads

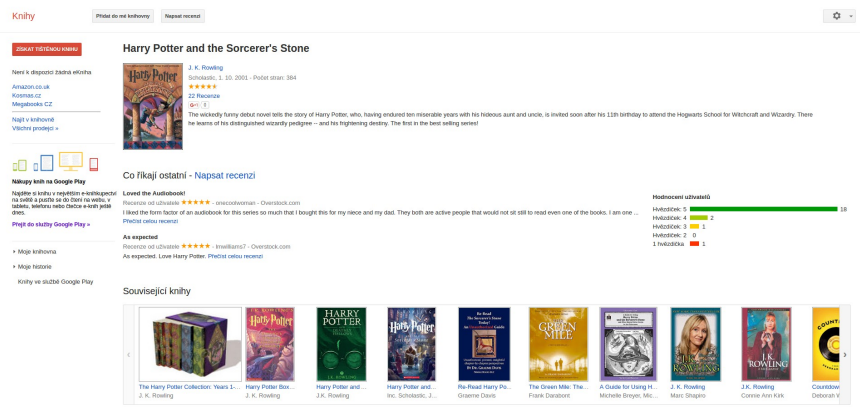
Portál tedy uživateli nabízí možnost vyhledávat knihy, třídit je podle stavu, přidávat je do vlastních poliček a komentovat jednotlivé tituly. Dále poskytuje informace o autorovi, jeho dalších dílech či doporučuje uživateli knihy podle dosavadního seznamu přečtených knih. Také umožňuje zobrazit informaci o době čtení knihy a disponuje systémem přátel.

2.2 Google Books

Google Books je vyhledávač společnosti Google zaměřený na knihy [5]. Dříve byla služba označena jako **Google Print** a jejím cílem bylo zpřístupnit digitalizované knihy knihoven veřejnosti[5] . Pro využívání služby není nutná registrace, nicméně některé z nabízených služeb jsou zpřístupněny až po přihlášení se do služby. Jelikož se jedná o službu společnosti Google, lze využít již existující Google účet. Služba je lokalizovaná pro Českou republiku a nabízí tedy uživatelské rozhraní v českém jazyce.

Bez přihlášení portál uživateli umožňuje vyhledávat knihy v katalogu a zobrazovat informace o nich. V detailu knihy je možné nalézt informace o vyhledaném titulu, mezi kterými jsou například počet stran titulu, nakladatelství či datum vydání. Nechybí zde ani komentáře přihlášených uživatelů. Na stránce je dále možné nalézt graf hodnocení knihy uživateli na stupnici 1–5 hvězdiček, kde 5 hvězdiček znamená největší spokojenost s titulem. V případě existence většího množství vydání hledaného titulu, je možné na stránce detailu najít i další vydání kromě právě navštíveného. Služba obdobně jako Goodreads navrhuje uživateli knihy, které s daným titulem souvisí. Zajímavostí detailu knihy je možnost generování citace knihy v různých formátech. Díky české lokalizaci je zde možné nalézt i odkazy k zakoupení knihy v českých obchodech.

Služba u knih, které byly nakladatelstvím či samotnými autory uvolněny ke zveřejnění, nabízí obsah knihy k volnému přečtení. Zveřejněny v plném rozsahu jsou i knihy, u kterých již vypršela ochrana autorským právem. U ostatních knih je možné najít úryvek z knihy, ovšem není to pravidlem [9].



Obrázek 2.2: Detail knihy Google Books

Přihlášenému uživateli je umožněno přidat vyhledanou knihu do několika seznamů, které představují knihovnu uživatele. Knihovna disponuje například seznamem přečtených, rozečtených či oblíbených knih. Jelikož může být účet propojen s aplikací **Google Play**[6] existuje zde i seznam knih zakoupených přes tuto aplikaci. Knihovna nabízí možnost přidání vlastních seznamů. Knihovna bohužel nebere na vědomí význam již předdefinovaných seznamů a tak je možné mít v jeden čas knihu jak v seznamu přečtených knih, tak v seznamu knih rozečtených, což není zrovna logické. Problém vzniká z důvodu pohlížení na seznamy jako na poličky, kde výskyt v několika poličkách naráz smysl dává. Jednotlivé seznamy je možné

[5] https://cs.wikipedia.org/wiki/Knihy_Google
[6] https://play.google.com/books

nastavit jako viditelné, případně skryté pro ostatní uživatele. Služba poskytuje možnost exportovat jednotlivé seznamy ve formátu **XML**.[7]

Bohužel jakákoli možnost sledování průběhu čtení knihy ve službě chybí. Datum přidání nebo odebrání knihy do/z jednotlivých seznamů chybí taktéž a není tedy možné ani přibližně zjistit, jak dlouho byla kniha čtena.

Výhodou Google Books je možnost částečně používat službu bez registrace. Přihlášený uživatel získává navíc možnost využití knihovny, tedy přidávání knih do jednotlivých seznamů. Kromě toho služba nabízí vyhledávání knih, zobrazení informací o nich, dohledávání jiných vydání požadované knihy, doporučuje knihy spojené s požadovanou knihou nebo podle seznamu oblíbených knih. Umožňuje knihy hodnotit a recenzovat. Dále nabízí odkazy k zakoupení knihy, a v některých případech, i možnost knihu číst bez nutnosti zakoupení.

2.3 Databazeknih.cz

Databáze knih je českou sociální sítí, která cílí na čtenáře knih [3]. Byla založena v roce 2008 Danielem Fialou[8]. Podle výroční zprávy **SČKN**[9] se jedná o nejnavštěvovanější knižní web v České republice [18]. V databázi knih je možné nalézt knihy vydané v českém nebo slovenském jazyce, případně knihy v jiném jazyce, ale vydané českým nebo slovenským nakladatelstvím na území ČR nebo Slovenska po roce 1945. Dále knihy zatím nevydané s termínem vydání do jednoho roku, u kterých je znám oficiální český nebo slovenský název, datum vydání a **ISBN**[10] [17].

Služba uživateli nabízí možnost vyhledávat tituly v rozsáhlé databázi knih, která je tvořena českými vydáními, čímž se výrazně liší od předešlých služeb, které nebyly přímo zaměřeny na Českou republiku. Vyhledávat tituly a zobrazovat jejich detailní informace je možné i bez registrace. V sekci detailu knihy jsou zobrazeny informace o vydání např. název díla, autor, datum vydání, nakladatelství či překladatel v případě cizojazyčné knihy. Zobrazeno je i hodnocení knihy registrovanými uživateli. Ke každé knize jsou dále přiřazeny štítky, díky kterým je možné shlukovat různé knihy do větších kategorií. Kromě informací o knize poskytuje detail i skrovné informace o autorovi a jeho další tvorbě. Na stránce nechybí ani odkazy ke koupi vybrané knihy.

Jak již bylo zmíněno, služba je sociální sítí, a tak disponuje i dalšími možnostmi oproti předešlým službám. V detailní sekci titulu je možné po přihlášení okomentovat titul, ale také hodnotit komentáře ostatních uživatelů. Dále je možné nahlédnout do zajímavostí knihy, vstoupit do diskuze či napsat recenzi. Nechybí zde ani možnost zobrazení dalších vydání vybrané knihy. Velkou zajímavostí je sekce *Bazar*, kde je možné odkoupit knihu od jiného uživatele nebo ji nabídnout k prodeji. Služba nabízí žebříčky knih podle různých kategorií. Po registraci nabízí možnost vytvářet seznamy knih pro ostatní uživatele např. ve stylu „*100 knížek, které by měl každý přečíst*“ apod. Existuje i možnost psát si veřejný blog či spravovat a přidávat knihy do databáze. Portál dále nabízí knižní novinky, soutěže či výzvy. Za zmínku stojí i detailní informace o jednotlivých autorech, kde je možné nalézt jeho biografii, tvorbu, ocenění či citáty. Nechybí ani možnost přidávat si přátele, tzv. oblíbené uživatele, se kterými je možné si dopisovat nebo sledovat jejich seznamy knih. Seznamy knih je možné upravovat či přidávat a existuje logika při přesouvání knih mezi některými

[7] https://cs.wikipedia.org/wiki/Extensible_Markup_Language
[8] http://www.danielfiala.cz/
[9] https://www.sckn.cz/
[10] https://cs.wikipedia.org/wiki/International_Standard_Book_Number

seznamy. Například je-li kniha v seznamu *Chystám se číst* a je přesunuta do seznamu, *Právě čtu* je automaticky odebrána ze seznamu *Chystám se číst*.

Otázka řešení sledování čtení knihy je zde řešena obdobně jako u Goodreads. Po přidání knihy do seznamu *Právě čtu* se uloží datum začátku čtení. Čtení knihy končí přesunutím knihy do seznamu *Přečtené*, čímž dojde k uložení data ukončení čtení knihy. Obě data je následně možné změnit u jednotlivých knih v seznamu *Přečtené*. Nicméně doba strávená čtením není opět nikde zobrazena.

Obrázek 2.3: Detail knihy Databazeknih.cz

Databazeknih.cz je sociální síť, která poskytuje velké množství českých vydání knih, mezi kterými uživatel může vyhledávat a následně zobrazovat informace o nich. Služba poskytuje další zajímavé možnosti, jako jsou například bazar knih, seznamy knih, žebříčky či diskuze nebo blogy. Po přihlášení se může uživatel účastnit těchto diskuzí, komentovat a hodnotit knihy nebo spravovat svoji vlastní knihovnu pomocí seznamů knih.

2.4 Shrnutí konkurenčních služeb

Služby spojené s vedením čtenářského deníku fungují v podstatě na stejném principu. Umožňují uživateli vyhledat knihu v databázi, zobrazit o ní stručné až detailní informace, okomentovat ji a přidat ji do nějaké formy seznamu. Mezi službami samozřejmě existují konkurenční odlišnosti. Následující podkapitoly vytyčí odlišnosti konkurenčních služeb vůči této aplikaci.

2.4.1 Goodreads

Goodreads splňuje většinu požadavků na aplikaci, nicméně zcela zde chybí možnost přidat literární rozbor ke knize. Možnost sledování četby je limitovaná pouze na začátek a konec čtení. Nelze stopovat jednotlivé úseky a výsledný čas strávený čtením knihy nelze nikde zobrazit. Je možné exportovat celou knihovnu ve formátu **CSV**[11], avšak tento formát není zcela vhodný. Goodreads oproti aplikaci nabízí možnost zobrazení informací o autorovi.

2.4.2 Google Books

Google Books nabízí vůči aplikaci možnost práce s autorem, zobrazení náhledu obsahu knihy nebo celého obsahu. Avšak zcela chybí možnost sledování průběhu četby, neexistuje zde ani záznam o tom, kdy byla kniha přidána do seznamu čtených nebo přečtených knih. Opět chybí možnost přidat literární rozbor. Export seznamů knih je zde sice dostupný ovšem ve formátu **XML**, který není pro běžného uživatele použitelný.

2.4.3 Databazeknih.cz

Databázeknih.cz nabízí například možnost práce s autorem, bazar, recenze či žebříčky, které aplikace nabízet nebude, avšak opět zde chybí možnost literárního rozboru. Služba poskytuje možnost zobrazení data začátku a konce čtení knihy, ovšem přístup není zcela přívětivý, navíc dobu strávenou čtením knihy je opět nutné ručně vypočítat. Export seznamu knih je možný pouze u přečtených knih ve formátu **XLSX**[12].

[11] https://cs.wikipedia.org/wiki/CSV
[12] https://cs.wikipedia.org/wiki/Office_Open_XML

Kapitola 3

Návrh aplikace

Tato kapitola se zaměřuje na návrh výsledné aplikace čtenářského deníku a je rozdělena do několika podkapitol. V podkapitole č. 3.1 jsou předloženy služby, které bude výsledná aplikace uživateli nabízet a aplikace je zde obecně popsaná. Podkapitola č. 3.2 se zaměřuje na cílové uživatele aplikace a jejich rozdělení. Navrhovaná architektura aplikace je nastíněna v podkapitole č. 3.3. Návrh uživatelského rozhraní aplikace včetně grafických podkladů je detailně popsán v podkapitole č. 3.4. Kapitolu uzavírá podkapitola č. 3.5, která se zabývá návrhem logické části aplikace.

3.1 Služby aplikace

V kapitole č. 2 byly vytyčeny některé body, které by měla aplikace pro vedení čtenářského deníku splňovat. V téže kapitole byly probrány i některé konkurenční služby, které ovšem některé body nesplňovaly.

Hlavní funkcí aplikace bude možnost vyhledávat knihy podle názvu titulu nebo jména autora. V případě, že je kniha úspěšně vyhledána, má být aplikace schopna zobrazit o této knize detailní informace. Mezi tyto informace patří název díla, jméno autora, datum vydání, název nakladatelství, popis díla a obálka knihy. Aplikace by měla být schopna zobrazit zdroj informací o titulu z více jak jednoho zdroje, mezi těmito zdroji má mít uživatel možnost přepínat. Je-li to možné, aplikace doporučí podobné knihy, na základě právě zobrazené knihy. Informace o vyhledané knize by měla doplnit možnost zakoupení knihy v některém z internetových obchodů. Detail knihy by měl také obsahovat komentáře ostatních uživatelů. Sekce by měla dále uživateli umožňovat přidat si knihu mezi své knihy nebo ji z nich odebrat. V případě, že se kniha nachází mezi uživatelovými knihami, má mít uživatel možnost přidat knihu do některé z poliček, změnit stav čtení knihy, případně začít nebo ukončit stopování úseku čtení knihy. Byla-li kniha již čtena nebo je-li čtena právě teď, je zobrazena tabulka jednotlivých úseků čtení s naměřenými časy a celkovým časem čtení. Sekce dále poskytuje možnost přidání literárního rozboru díla v případě, že daná kniha se nachází mezi uživatelovými knihami. Nechybí možnost exportovat detail knihy včetně zmíněného literárního rozboru ve formátu **PDF**[1].

Zmíněnou možnost stopování doby čtení knihy bude možné obsluhovat ze sekce detailu knihy a ze stopovacího menu v hlavním navigačním panelu. Intervaly naměřené pomocí této služby se promítnout v sekci přehledu čtení.

[1] https://cs.wikipedia.org/wiki/Portable_Document_Format

Další funkcí aplikace bude moci zobrazit si všechny knihy, které si uživatel přidal do své knihovny. Knihy bude možné v tomto náhledu filtrovat podle jejich stavu (přečtené, chci přečíst apod.) a vyhledávat mezi nimi. Uživatel bude mít možnost v této sekci exportovat seznamy knih ve formátu **PDF** a **TXT**[2]. Exportovat je možné všechny knihy, případně exportovat knihy podle zvoleného stavu.

Knihy bude moci uživatel přidávat do poliček. K práci s těmito poličkami bude existovat speciální sekce, ve které bude možné poličku vytvořit. Již existující poličku bude možné smazat nebo jí editovat ve formě změny jména. Nicméně přidávání či odebírání knih zde nebude možné, jelikož k tomu slouží sekce detailu knihy.

Aplikace bude také poskytovat možnost zobrazit si přehled čtení ve speciální sekci. V této sekci budou uvedeny informace o celkovém počtu knih uživatele, dále také o počtu knih přečtených, čtených, knih k přečtení a knih zatím nepřečtených. Přehled bude také obsahovat celkový fyzický čas strávený čtením všech knih. Průběh čtení bude uživateli nabídnut ve formě dvou časových os. Jedna z os bude zobrazovat intervaly čtení čtených knih za vybraný měsíc. Druhá osa bude zobrazovat dobu čtení všech knih uživatele od doby registrace.

Uživateli budou nabízeny nedávno přidané knihy do databáze aplikace, které jej mohou inspirovat k četbě knih neznámého autora či k přidání některé z této knih do seznamu knih k přečtení. Seznam těchto knih se bude automaticky aktualizovat.

Nabídnuta bude také sekce se správou osobního profilu, ve kterém bude možné provést změnu profilového obrázku či osobních údajů. Tato sekce bude obsahovat podsekci přátel, která uživateli umožní vyhledávat přátele mezi existujícími uživateli aplikace a posílat jim žádosti o přátelství. K vidění zde budou odeslané žádosti o přátelství, které bude moci zrušit, příchozí žádosti o přátelství od ostatních uživatelů, na které bude možné reagovat přijmutím či odmítnutím žádosti a samotný seznam přátel s možností tyto přátele odebrat. Po vybrání přítele se uživateli zobrazí jeho detailní profil včetně osobních údajů, doby přátelství a výčtu knih a poliček.

V neposlední řadě aplikace nabídne uživateli možnost využít registrace a přihlašování pomocí sociálních sítí **Facebook**[3] a **Google+**[4]. Kromě těchto možností přihlášení bude aplikace podporovat i ověření emailem a heslem.

3.2 Cíloví uživatelé

Aplikace cílí na dvě skupiny uživatelů. První skupinou jsou studenti, kteří si v rámci studií vedou čtenářský deník, v němž je podmínkou vypracovat literární rozbor díla. Mezi takové studenty se dají zařadit například studenti připravující se k maturitním zkouškám, vysokoškolští studenti literatury či učitelé na základních a středních školách. Druhou cílovou skupinou jsou lidé, kteří rádi čtou a chtějí mít přehled o svých knihách. Vedou si seznamy knih, které už přečetli nebo které by si rádi v budoucnu přečetli. Této skupině uživatelů jistě přijde vhod možnost sledovat průběžné čtení knih a zobrazení knih na časové ose. Prolínání těchto dvou skupin je samozřejmě možné a žádoucí pro využití plného potenciálu aplikace.

Z důvodu neznámého věkového zastoupení uživatelů a jejich praktických zkušeností s počítači je nutné vytvořit takové grafické uživatelské rozhraní, které bude jednoduché, přehledné a intuitivní k používání.

[2] Běžný textový soubor. Seznam nebude obsahovat obálku knihy.
[3] https://www.facebook.com/
[4] https://plus.google.com/

3.3 Architektura aplikace

Aplikace bude využívat architektury typu **klient-server**. Jedná se o architekturu, ve které existuje **server**, který čeká na požadavky od **klienta** a na základě typu požadavku odpovídá daty, jež klient následně zobrazuje. Při této architektuře existuje většinou jeden server a více klientů, které server konkurentně obsluhuje [19]. Jednou z výhod této architektury je odklonění zátěže z klienta na server, jelikož server připravuje data, které klient pouze zobrazí, neberou-li se v potaz drobné validace dat na straně klienta před odesláním požadavku, mezi než může patřit například ověření hodnot položek ve formulářích. Takový klient se označuje jako **tenký klient** [21]. Další výhodou je sdílení dat mezi více uživateli. K serveru se přistupuje pomocí **API**[5], díky kterému je možné k serverové části případně přistupovat i z dalších klientů, což umožňuje vývoj klienta na jiné platformě při zachování stejné serverové části.

Existuje i několik dalších typů klientů např. **tlustý klient**, který se liší především svojí soběstačností. Tento typ klienta nevyžaduje nutně serverovou část, jelikož se k databázi aplikace může připojovat sám a tak získávat samostatně potřebná data, které také samostatně zpracovává před jejich zobrazením. Tlustý klient většinou vyžaduje instalaci na klientský počítač a je hůře udržovatelný. Z tohoto důvodu bude v této práci klient navržen ve formě **tenkého klienta**.

3.4 Návrh grafického uživatelského rozhraní klienta

V této podkapitole bude popsán návrh uživatelského rozhraní klientské části aplikace elektronického čtenářského deníku.

Aplikace má dva hlavní stavy. Prvním ze stavů je okno, které uživatel uvidí při otevření aplikace, kde bude vyzván k registraci či přihlášení do systému viz. obrázek č. 3.1. Druhým stavem je okno, které uživatel uvidí až poté, co je úspěšně přihlášen a bude v sobě obsahovat veškeré podsekce aplikace. Návrh okna je možné vidět na obrázku č. 3.2.

Obrázek 3.1: Návrh přihlašovacího okna

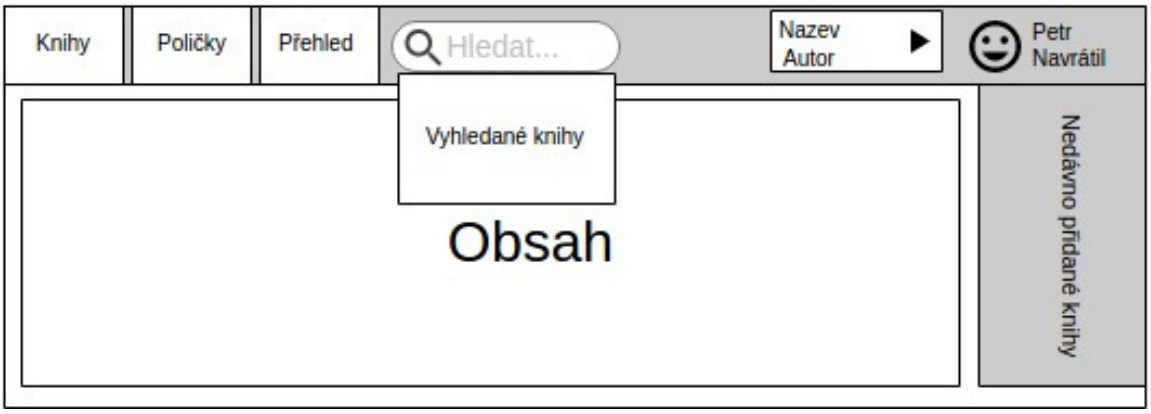
Hlavní okno aplikace je rozděleno na tři části. Nejhornější částí je navigační panel, pomocí kterého se bude uživatel moci pohybovat po aplikaci. Panel dále obsahuje vyhledávací pole, které slouží k vyhledávání knih v databázi pomocí názvu knihy nebo jména autora. V případě nalezení knih se objeví vysouvací okno se seznamem nalezených knih, kde kromě obálky knihy bude zobrazen i název díla a jméno autora. Po kliknutí na vyhledanou knihu

[5] https://cs.wikipedia.org/wiki/API#Web_API

je uživatel přesměrován do detailní sekce této knihy, kde s ní může dále pracovat. Další položkou panelu je tzv. „*tracker*“, který se zobrazí v případě, že uživatel stopoval či aktuálně stopuje nějakou knihu a umožní mu okamžitě ve stopování pokračovat nebo stopování ukončit. Kliknutím na tracker se uživatel dostane do detailní sekce knihy, se kterou je stopování spjato. Poslední částí panelu je avatar[6] a uživatelské jméno přihlášeného uživatele. Kliknutím na tuto část je uživatel přesměrován do sekce, která se stará o uživatelský profil a systém přátel. Panel je fixovaný k hornímu rámu obrazovky, aby bylo možné okamžitě změnit zobrazovanou sekci.

Další sekcí je panel v pravé části, který zobrazuje posledních n přidaných knih do databáze. Po kliknutí na knihu je uživatel přesměrován do detailní sekce této knihy. Panel je fixovaný k pravé části okna a nehýbe se zbytkem okna, jelikož disponuje vlastním posuvníkem.

Poslední částí okna je plocha vyhrazená pro obsah, jež se bude dynamicky měnit podle aktuálně zobrazované sekce uživatelem.



Obrázek 3.2: Návrh hlavního okna aplikace

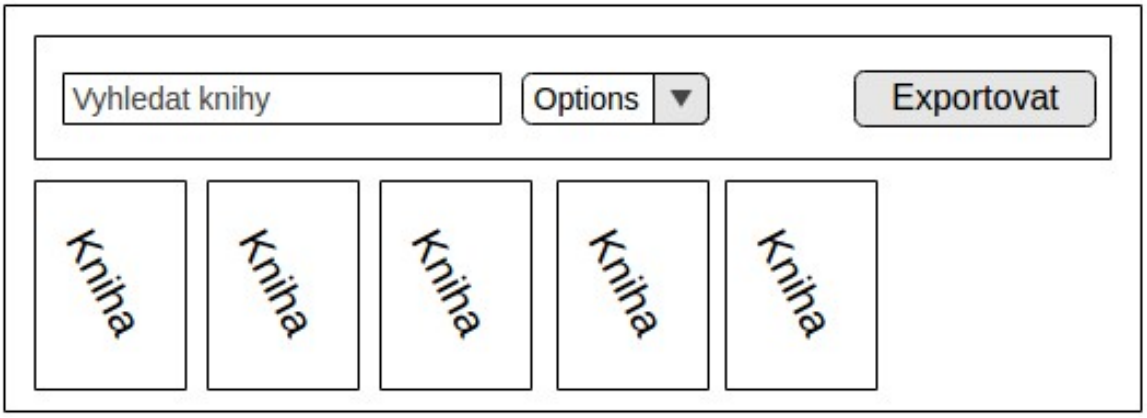
3.4.1 Knihy

Na obrázku č. 3.3 je možné vidět návrh grafického rozhraní pro sekci knih uživatele. Sekce je rozdělena na dvě části. Do první části patří vyhledávací okno, které pomocí zadaného klíče filtruje knihy. Filtrování je možné zpřesnit pomocí vysouvací nabídky, která se nachází vedle vyhledávacího okna a umožňuje zvolit zobrazení např. jen přečtených knih. Sekce obsahuje tlačítko pro exportování seznamu knih podle zvoleného výběrů vysouvací nabídky. Druhou část sekce tvoří jednotlivé knihy uživatele viz. obrázek č. 3.14. Po kliknutí na knihu je uživatel přesměrován do detailní sekce knihy.

3.4.2 Poličky

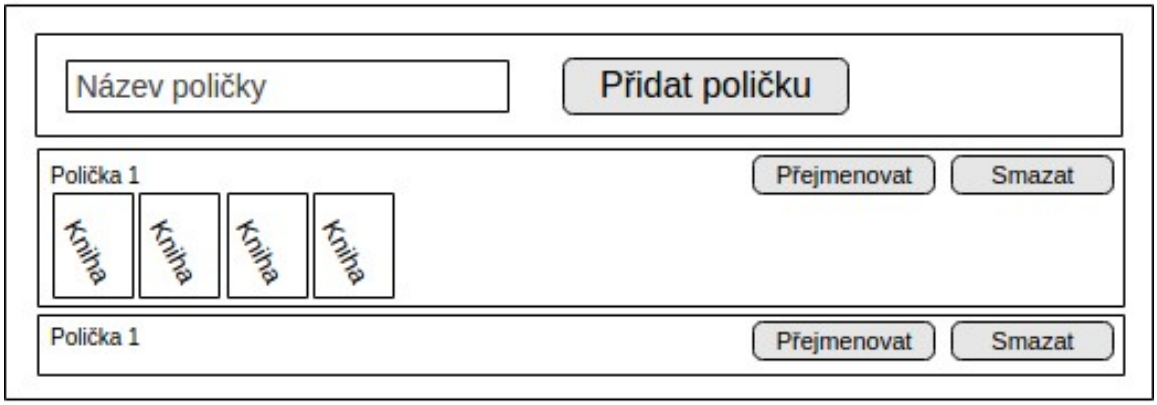
Na obrázku č. 3.4 je zobrazen plánovaný návrh grafického rozhraní sekce poliček uživatele. Sekce je opět rozdělena do dvou částí. V první polovině sekce se nachází textové okno, do kterého je možné vepsat název nové poličky a poličku následně přidat pomocí tlačítka po pravé straně. Druhou sekci tvoří samotné poličky. Jelikož poličky mohou shromažďovat velké množství knih, je polička koncipovaná jako vysouvací prvek z důvodu úspory místa v okně.

[6] Uživatelský obrázek.



Obrázek 3.3: Návrh sekce knih

V zavřeném stavu prvku poličky, je viditelné pouze její jméno, které je možné upravovat v případě, že se nepřekrývá s názvem jiné poličky uživatele. Poličku je také možné smazat pomocí speciálního tlačítka. Po kliknutí na prvek poličky se ze spodní části vysune zbývající obsah poličky, který představuje knihy do ní patřící. Návrh knihy je zobrazen na obrázku č. 3.14. Po kliknutí na jednu z knih bude uživatel opět přesměrován do detailní sekce této knihy.



Obrázek 3.4: Návrh sekce poliček

3.4.3 Přehled čtení

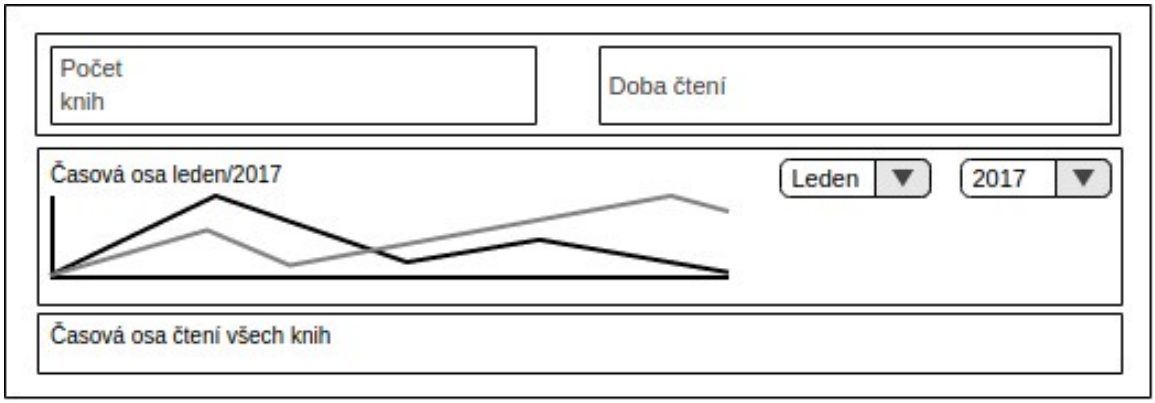
Na obrázku č. 3.5 je možné vidět návrh rozhraní pro sekci přehledu čtení uživatele. Sekce je rozdělena do tří částí. První část sekce tvoří informace o knihách uživatele, ve kterých lze nalézt celkový počet knih uživatele, počet přečtených knih apod. Také lze v této sekci zjistit, kterou knihu uživatel přečetl nejvíckrát, případně kolik času strávil čtením knih obecně. Zbývající dvě sekce jsou velice podobné a obsahují časové osy průběhu čtení uživatele za určitá období.

První ze zbývajících sekcí obsahuje časovou osu, na které je zachycen průběh čtení knih ve zvoleném měsíci. Průběh zde představují jednotlivé intervaly úseků čtení knihy. Z osy bude tedy možné zjistit, kolikrát jsme četli danou knihu ve zvoleném období a po

jak dlouhou dobu. Měsíc je možné změnit pomocí výsuvné nabídky v pravé části sekce, obdobně lze změnit i rok sledovaného měsíce.

Poslední sekcí přehledu čtení je část s osou celkového přehledu přečtených knih. Lze zde vidět časové zobrazení jednotlivých čtení knih od registrace uživatele, přičemž zobrazený úsek tvoří začátek a konec čtení knihy.

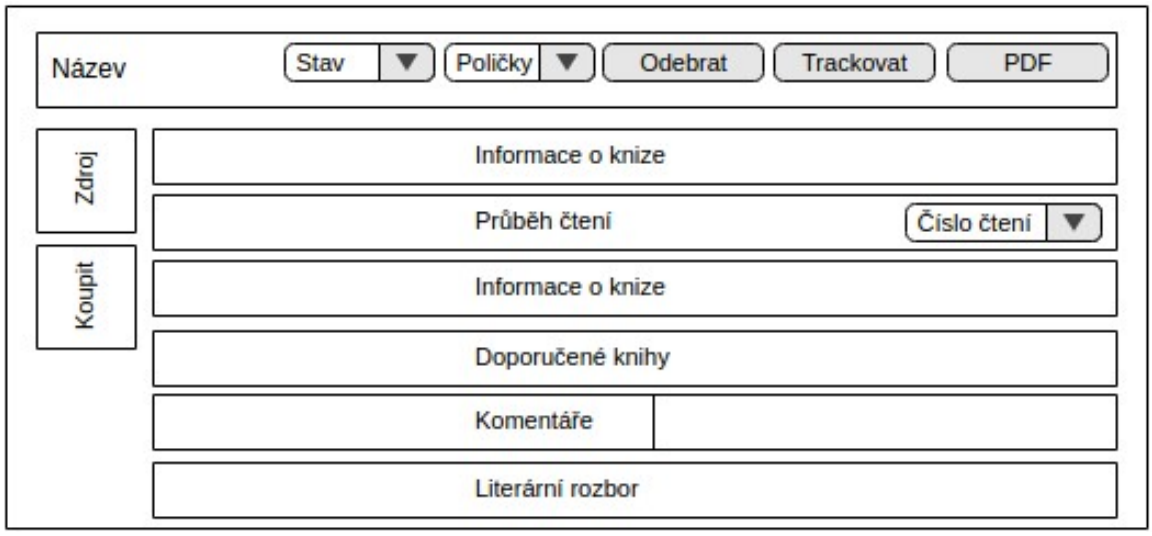
Obě časové osy jsou umístěné ve vysouvacích prvcích z důvodu úspory místa a přehlednosti, jelikož knih se mohlo od registrace uživatele nastřádat mnoho. Kliknutím na hlavičku prvku dojde k vysunutí/zasunutí obsahu prvku, tedy vlastní osy.



Obrázek 3.5: Návrh sekce přehledu čtení

3.4.4 Detail knihy

Na obrázku č. 3.6 je možné vidět návrh detailu knihy, který tvoří největší část celé aplikace. Sekce je logicky rozdělena do několika menších podsekcí. První sekce obsahuje název navštívené knihy a několik tlačítek, pomoci kterých je možné s knihou interagovat.



Obrázek 3.6: Návrh sekce detailu knihy

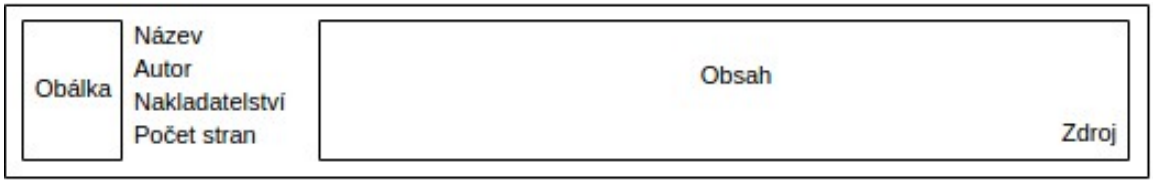
Na obrázku návrhu lze vidět všechna tlačítka, ovšem tlačítka se objevují a mění podle aktuálního stavu knihy. Není-li kniha mezi knihami uživatele jsou uživateli nabídnuta pouze

tlačítka **přidat knihu, stav a poličky**, přičemž s posledními dvěma nelze interagovat. Po přidání knihy mezi knihy uživatele se tlačítko **přidat** změní na tlačítko **odebrat**. Tlačítka **stav a poličky** začnou být aktivní a přibudou tlačítka **stopovat a pdf**, která umožní vygenerovat detail knihy a začít stopovat úsek čtení knihy. V případě, že je kniha aktuálně čtena, tlačítko **stopovat** ukončí stopování úseku.

V levé části okna lze vidět podsekci zdroj, která slouží k výběru zdroje informací o titulu. Podsekce bude koncipovaná do formy tlačítek, kterými se bude výběr přepínat. Níže od této podsekce se nachází část s odkazy na některé internetové obchody, kde je knihu možné zakoupit.

Informace o knize

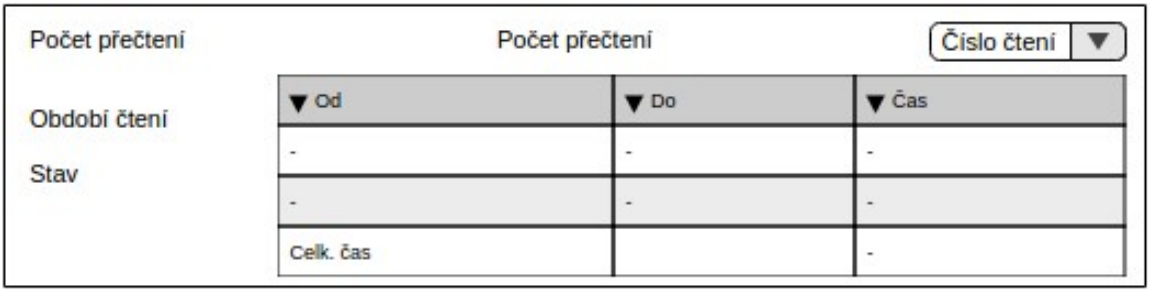
Podsekce informace o knize obsahuje název knihy, jméno autora, nakladatelství a počet stran. Zobrazen je také obsah knihy v podobě krátkého popisu. Podsekce obsahuje i název zdroje, ze kterého informace pocházejí. Po kliknutí na tento zdroj je uživatel přesměrován na originální stránku zdroje.



Obrázek 3.7: Návrh části informace o knize

Průběh čtení

Podsekce průběh čtení zobrazuje úseky čtení knížky. Je zde zobrazen celkový počet přečtení knihy a celková strávená doba čtením. Pomocí tlačítka výběru čtení je možné zobrazit si detailní tabulku intervalů vybraného čtení. Nalevo od tabulky se nacházejí stav, ve kterém se kniha nalézá, jelikož je možné zobrazit intervaly i aktuálně čtené knihy, a také počáteční a koncový datum čtení. Samotná tabulka disponuje datem začátku a konce čtení úseku knihy a z těchto dat spočtenou dobu čtení. Poslední řádek tabulky zobrazuje celkový čas strávený čtením knihy v tomto běhu.



Obrázek 3.8: Návrh části průběh čtení

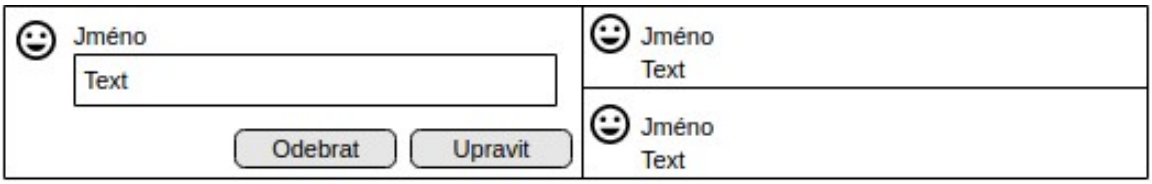
Doporučené knihy

Podsekce doporučené knihy je koncipována jako pás knih, které mají stejný vzhled jako ve zbytku aplikace viz. obrázek č. 3.14, kterým lze posouvat pomocí levého a pravého tlačítka. Po kliknutí na knihu je obsah detailu knihy aktualizován podle této knihy.

Obrázek 3.9: Návrh části doporučené knihy

Komentáře

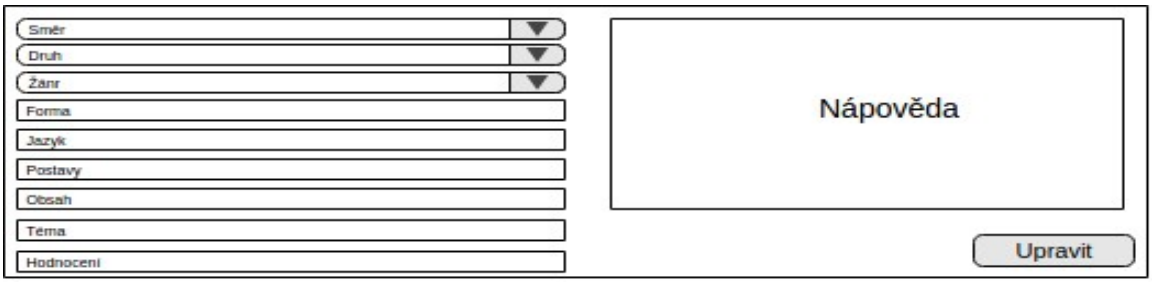
Podsekce komentáře je rozdělena do dvou částí. Levá část je vyhrazena přihlášenému uživateli a umožňuje mu přidat komentář ke knize. V případě, že knihu již komentoval, může tento komentář smazat nebo upravit. Druhá sekce je vyhrazena ostatním uživatelům a jsou zde vidět jejich komentáře. Kromě textu komentáře lze v této sekci najít jméno i avatar autora komentáře včetně data přidání.



Obrázek 3.10: Návrh části komentáře

Literární rozbor

Poslední podsekcí sekce detailu knihy je literární rozbor, který je rozdělen do dvou částí. První část slouží k vyplnění literárního rozboru pomocí vysouvacích nabídek nebo textových boxů. Druhou část tvoří nápověda k jednotlivým částem rozboru, která se aktivuje najetím kurzoru myši na prvek literárního rozboru. U možností vysouvací nabídky nápověda poskytuje základní informace o jednotlivých možnostech a opět se aktivuje najetím myši na požadovanou možnost. Pod nápovědou si lze představit například charakteristiku literárního směru v případě najetí myší na některý z literárních směrů.



Obrázek 3.11: Návrh části literární rozbor

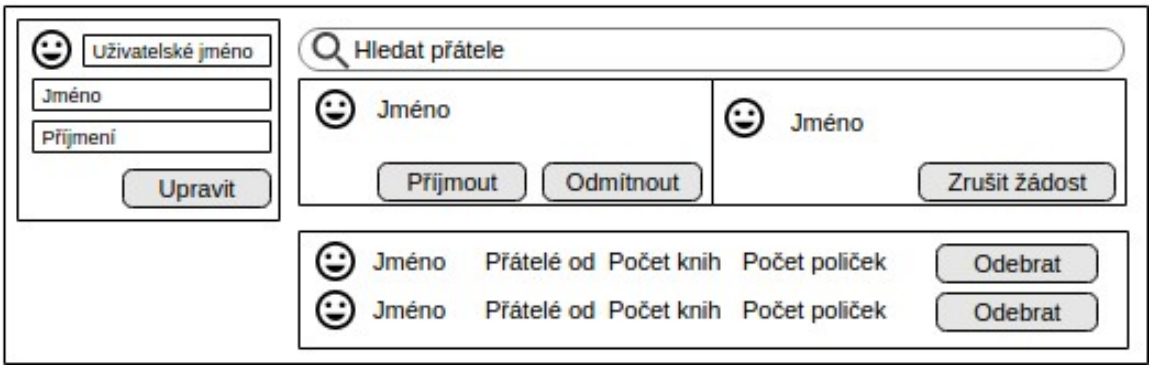
3.4.5 Profil uživatele

Na obrázku č. 3.12 je zobrazen návrh profilu uživatele. Tato sekce je rozdělena do několika částí, které budou postupně popsány.

Nejlevější částí je část samotného uživatele, kde je možné upravovat osobní údaje, jakožto uživatelské jméno, jméno, příjmení a avatar uživatele. Napravo od této části je možné nalézt textové okno, které slouží k vyhledávání uživatelů v databázi. Po vyhledání se uživateli zobrazí seznam nabízených uživatelů včetně tlačítek pro interakci s tímto uživatelem. Lze odtud poslat například žádost o přátelství případně přátelství zrušit.

Následující část se zabývá žádostmi o přátelství uživatele. V levá části je možné vidět příchozí žádosti o přátelství, které lze pomocí tlačítek přijmout či odmítnout. Pravá část zobrazuje žádosti odeslané uživatelem a lze je pomocí tlačítka zrušit.

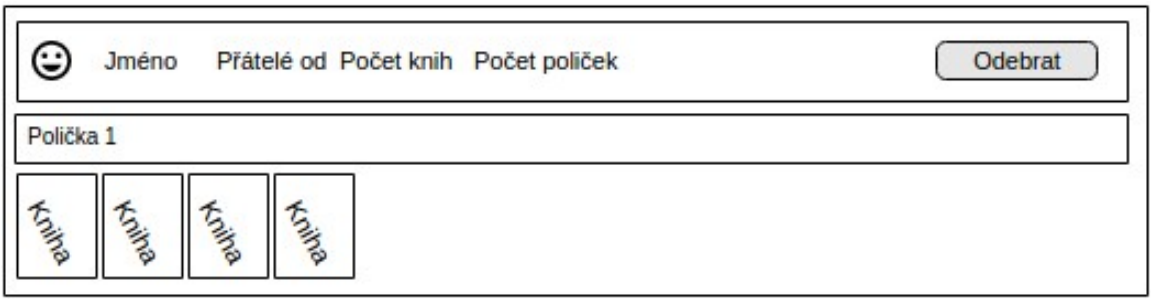
Poslední částí sekce uživatelského profilu je přehled přátel. Jedná se o seznam, který obsahuje avatar uživatele, jméno, datum začátku přátelství a počet knih i poliček. V této sekci je možné přátelství s uživatelem ukončit pomocí dostupného tlačítka. Po kliknutí na položku přítele v seznamu, je uživatel přesměrován do profilu vybraného přítele.



Obrázek 3.12: Návrh sekce profilu

3.4.6 Profil přítele

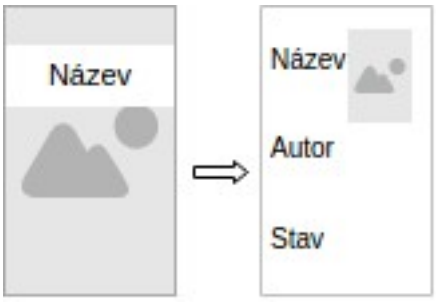
Profil přítele je zobrazen na obrázku č. 3.13. První část tvoří informace o příteli, které jsou graficky navrhnuty stejně jako v podkapitole č. 3.4.5. Druhou část sekce tvoří seznam poliček přítele. Návrh je opět stejný jako v kapitole č. 3.4.2. Poslední část tvoří výčet všech knih uživatele viz obrázek č. 3.14.



Obrázek 3.13: Návrh sekce profilu přítele

3.4.7 Element knihy

Na obrázku č. 3.14 je možné vidět návrh elementu knihy. Tento element je v aplikaci použit několikrát a jeho vzhled je pokaždé stejný. Element knihy má dva stavy. Klidový stav zobrazuje pouze název knihy a obálku knihy, která zabírá celou plochu elementu. Nejetím kurzoru myši na element knihy se karta dostává do druhého stavu, ve kterém je zobrazen název knihy, jméno autora a stav knihy vůči uživateli. Obálka knihy je zmenšena do pravého horního rohu elementu.



Obrázek 3.14: Návrh sekce profilu přítele

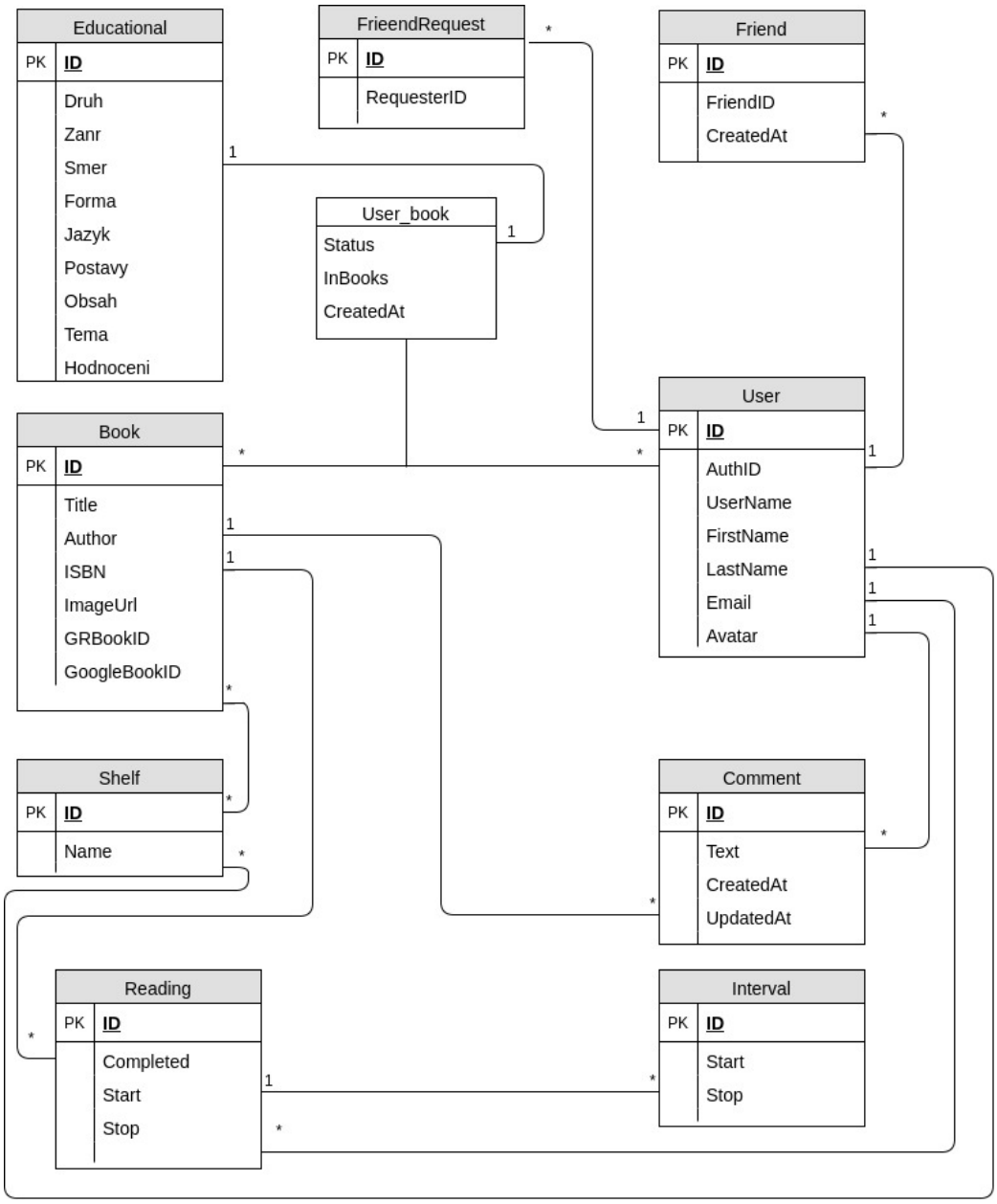
3.5 Návrh serverové části

Tato kapitola se zabývá návrhem logické části aplikace čtenářského deníku, tedy návrhem serveru. Jelikož se jedná o nepostradatelnou část aplikace, která obstarává data pro klientskou část, je nutné, aby obě části aplikace spolu dokázaly správně komunikovat a poskytovat požadované služby. Komunikace mezi částmi aplikace bude zajištěna pomocí protokolu **HTTP**[7] či zabezpečené verze **HTTPS**[8].

Je nutné zajistit, aby serverová část uživateli umožňovala registrovat se nebo se přihlásit k používání aplikace. Jelikož je zmíněna možnost přihlašovat se pomocí sociálních sítí, bude k zajištění této služby využita třetí strana tzv. **Auth0**[9] více kapitola č. 4.3, jež se touto problematikou zabývá a umožňuje tuto možnost integrovat do dalších aplikací. Aby bylo možné rozlišit jednotlivé uživatele v této části aplikace, bude při komunikaci mezi zmíněnými částmi využit tzv. **JWT token**[10] viz. kapitola č. 4.4, který slouží ověření identity odesílatele požadavku na straně serveru při užití bezstavové komunikace. Server bude nabízet několik **rout**[11], na které bude klient posílat HTTP požadavky. Na jejich základě dojde k transformaci dat, která vyústí v odpověď klientovi. Routy budou logicky členěny podle skupiny dat k transformaci např. **books, users** apod. Princip jednotlivých rout a dalších služeb aplikace je více popsán v kapitole č. 4.6, která se zabývá vlastní implementací serverové části čtenářského deníku.

[7] https://tools.ietf.org/html/rfc2616
[8] https://tools.ietf.org/html/rfc2818
[9] https://auth0.com/
[10] https://jwt.io/
[11] Routa je část adresy, na kterou se posílá požadavek při komunikaci klienta se serverem a zastupuje logický celek, se kterým se má provést požadovaná změna definovaná typem požadavku. Např. při zaslání požadavku **DELETE** na adresu *http://www.diary-server.cz/books/:id*, lze očekávat, že kniha specifikovaná pomocí *id* bude smazána z kolekce knih.

Jelikož je nutné ukládat nepřeberné množství informací o jednotlivých uživatelích, jejich přátelích, knihách či čtení těchto knih, bude tato část využívat práci s databází. Návrh databáze je možné vidět níže na obrázku č. 3.15, kde je návrh znázorněn pomocí **Entity relationship diagramu**[12].



Obrázek 3.15: Entity relationship diagram databáze

[12]https://cs.wikipedia.org/wiki/Entity-relationship_model

Kapitola 4

Implementace

Tato kapitola se zaměřuje na vlastní implementaci aplikace čtenářského deníku a lze zde také najít popsané technologie či služby, které byly využity k její implementaci. Kapitola se dělí na několik podkapitol, které popisují jednotlivé technologie a v poslední řadě i vývoj klientské části aplikace a serverové části aplikace.

Pro splnění podmínky multiplatformní aplikace byla zvolena cesta implementovat aplikaci jako aplikaci webovou, čímž se zajistí dostupnost aplikace na velké škále platforem. Již dříve zmíněná architektura klient-server navíc umožní případný vývoj platformě specializované aplikace do budoucna. Technologií pro vývoj moderních, dynamických, webových aplikací, ve kterých je umožněna velká interakce s uživatelem, existuje mnoho, jmenovitě např. **PHP**[1], **React**[2], **Angular 2**[3] a další. PHP se pro implementaci klient-server architektury nehodí, jelikož se výsledná stránka sestaví již na straně serveru, a klient tedy není potřeba, jedná se o tzv. **server-side rendering**. Další nevýhodou je aktualizace obsahu stránky, která v tomto případě končí aktualizací celé stránky, jelikož je nutné ji znovu celou sestavit na straně serveru. Opačným přístupem je tzv. **client-side rendering**, kdy je stránka sestavena až na straně klienta ze zdrojů poskytnutých serverem. Sestavení stránky zajistí jazyk **JavaScript**[4], který je vykonán prohlížečem [20]. Tento přístup přináší velkou výhodu při aktualizaci obsahu stránky, jelikož stačí požádat o nová data a po přijetí dat příslušný obsah překreslit. Zbylé dvě zmíněné technologie využívají client-side renderingu a vzájemně mezi sebou soupeří v konkurenčním boji. Nicméně pro vývoj klientské části aplikace byl zvolen Angular 2 z důvodu předešlých zkušeností s touto technologií. Angular 2 je blíže popsán v kapitole č. 4.1.

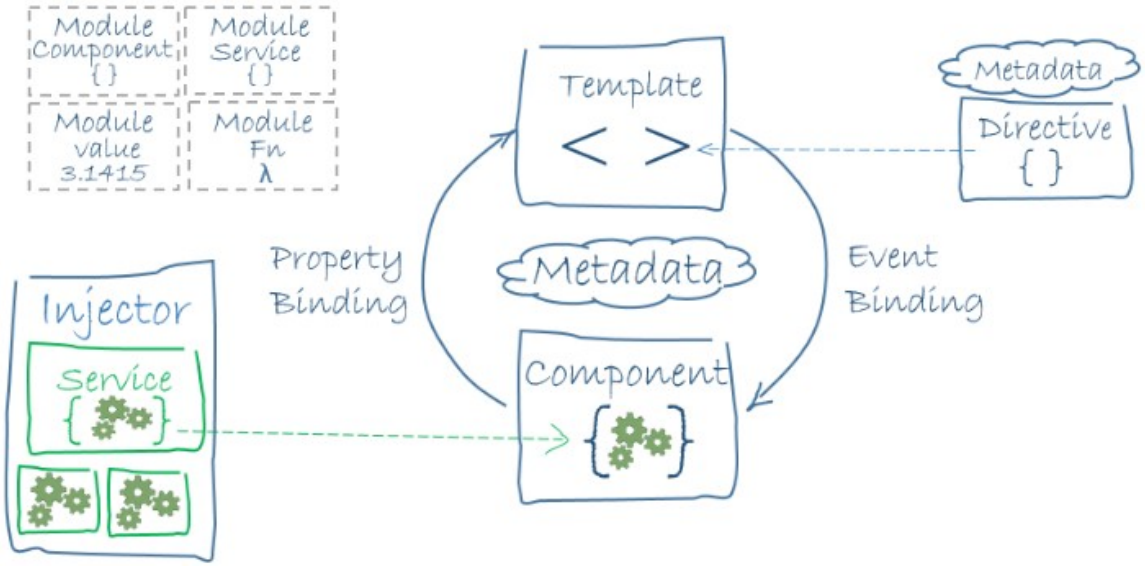
Pro serverovou část byl zvolen programovací jazyk **Go** nebo-li **Golang**[5], kvůli syntaxi odvozené z jazyka **C** a existenci mnoha balíčků usnadňující práci s databází, http dotazy apod. Jazyk je více přiblížen v kapitole č. 4.2. Kapitola č. 3.5 se zmiňuje o využití služby **Auth0** pro autorizaci a autentizaci uživatele a tvoří důležitou složku aplikace, více informace o této službě je možné nalézt v kapitole č. 4.3. Se službou je spojený i **JWT token**, který je přiblížen v kapitole č. 4.4. Implementace klienta pomocí Angularu 2 je popsaná v kapitole č. 4.5 a implementace serveru v kapitole č. 4.6.

[1] https://secure.php.net/
[2] https://facebook.github.io/react/
[3] https://angular.io/
[4] https://www.javascript.com/
[5] https://golang.org/

4.1 Angular 2

Angular 2 je open source projekt, jehož největším přispěvatelem je společnost Google. Jedná se o javascriptový framework, který cílí na vývoj tzv. **single-page**[6], dynamických aplikací. Dalším z cílů frameworku je možnost využít tentýž kód pro více způsobů distribuce aplikace, například ve formě webové stránky či desktopové aplikace [1]. Vývoj aplikace probíhá za pomocí jazyků **HTML, CSS** a jazyku **JavaScript**, případně jazyku, který je do JavaScriptu možné přeložit např. **TypeScript**.

Aplikace vytvořená pomocí tohoto frameworku se skládá z mnoha menších částí viz. obrázek č. 4.1. Hlavním prvkem architektury je tzv. **komponenta**, která zastupuje nějaký fyzický element obsažený v aplikaci. V případě čtenářského deníku si lze pod komponentou představit například element knihy, viz. kapitola č. 3.14, který se v aplikaci vyskytuje několikrát. Vzhled této komponenty je popsán v **template**, který je spojen s touto komponentou a zobrazuje data poskytnutá komponentou. Interakce uživatele s komponentou může vyvolat událost, na kterou komponenta definovaným způsobem reaguje. V případě potřeby změny dat, například odstranění knihy z poličky, komponenta využije dalšího prvku frameworku, a to tzv. **servisu**, který slouží např. ke komunikaci mezi komponentami nebo ke komunikaci klietské aplikace se serverovou částí [13]. Lze tedy říci, že framework využívá tzv. **Model-View-Controller** architektury[7]. Model je v tomto případě reprezentován servisy, view templaty a controller komponentami. Komponenty je možné zanořovat do sebe, a tím tvořit komplikovanější komponenty, např. komponenta uživatelských poliček, která obsahuje komponenty poliček, které obsahují komponenty knih. Stejný jev lze pozorovat i u servisů, jež je také možné využít pro vytvoření dalších servisů.



Obrázek 4.1: Nákres Angular 2 architektury; Zdroj nákresu Angular 2 [8]

.

[6] Aplikace má pouze jednu stránku, která se pomocí Javascriptu dynamicky mění, ačkoli se může zdát, že adresa zobrazené stránky se mění.

[7] https://cs.wikipedia.org/wiki/Model-view-controller

Dalším nezbytným prvkem frameworku je tzv. **router**, který se stará o navigaci v aplikaci. Disponuje tabulkou platných adres[8], které jsou spojeny s komponentou, která se má zobrazit v případě, že se uživatel nachází na některé z platných adres. V případě, že se uživatel ocitl na stránce s neplatnou adresou, je zobrazena defaultní komponenta, např. přihlášení do aplikace. Router ke své činnosti potřebuje vědět, kam má požadovanou komponentu na stránce zobrazit, k čemuž slouží speciální **HTML TAG**[9]. Tyto tagy je též možné částečně zanořovat. Při změně adresy je z **DOM**u[10] rekurzivně odebrána komponenta spjata s předešlou adresou a nová komponenta je umístěna místo ní včetně vytvoření všech podkomponent či servisů.

Framework dále nabízí několik speciálních **strukturálních direktiv**[11], které umožňují jednodušší návrh templatu komponent. Mezi tyto direktivy patří např. `*ngIf="výraz"`, která v případě pravdivého booleovského výrazu přidá element do DOMu, v případě nepravdivého jej odebere. Hojně užívaná je i direktiva `*ngFor = "let x of array"`, která umožní zobrazit element tolikrát, kolik prvků obsahuje kolekce `array`. Obdobným způsobem je možné měnit CSS property elementů nebo jim dynamicky upravovat přiřazené CSS classy[12]. Framework také usnadňuje práci s CSS animacemi a přechody.

4.2 Golang

Golang nebo Go je open source programovací jazyk vytvořený společností Google, který byl představen v roce 2009 [7]. Golang je kompilovaný jazyk, který je staticky typovaný nicméně disponuje i vlastnostmi dynamicky typovaného jazyka, jako je možnost odvození typu proměnné během její inicializace [14]. Jazyk disponuje vlastním balíčkovacím systémem, který umožňuje pohodlně využívat balíčky jiných vývojářů.

Struktura golang kodu je členěna na **moduly**, přičemž za modul se považuje podadresář hlavního adresáře. Veškeré `.go` soubory v adresáři modulu automaticky patří do tohoto modulu. Moduly se navzájem nevidí, je tedy nutné je do ostatních modulů importovat. Golang velice dbá na čistotu kódu, a tak vyžaduje dodržení některý pravidel, mezi která patří například absence nevyužitých importů či proměnných. Také je nutné dodržet některá z pravidel formátování zdrojového kódu. Některá z pravidel umí jazyk automaticky aplikovat, a tím zformátovat kód do požadované podoby. V případě nesplnění některého z pravidel nelze zdrojový kód přeložit.

4.3 Auth0

Auth0 je služba, která poskytuje řešení autorizace a autentizace uživatele vůči aplikaci. Hlavní výhodou této služby je možnost využít již existujících účtů různých sociálních sítí či služeb třetích stran, jako je například Facebook či Github [2]. Jelikož každá z těchto třetích stran využívá rozdílných způsobů autentizace, bylo by komplikované pro každou z požadovaných třetích stran zvlášť implementovat způsob, jak autentizace využít v cílové aplikaci. Na tento problém se Auth0 soustředí a poskytuje jednodušší řešení, jak s autentizací pomocí třetích stran pracovat.

[8] Příklad řádku tabulky: `path: "platform/shelves", component: ShelvesComponent`

[9] Značka v Markup jazyce. V tomto případě `<router-outlet>`.

[10] Document Object Model

[11] Dynamicky mění obsah DOMu.

[12] Identifikátor, ke kterému jsou v souboru CSS této komponenty napsaná pravidla, které jej graficky upravují.

Služba nabízí jednotné API, které zastřešuje API všech nabízených třetích stran, čímž usnadňuje možnost jejich užití v nové aplikaci. Různé účty uživatele je navíc možné spojovat, a tím umožnit uživateli přístup k účtu aplikace pomocí více různých účtů. Kromě možnosti užití třetích stran, služba samozřejmě nabízí i možnost přihlašování se pomocí hesla a emailu.

Není-li stanoveno jinak, je nově registrovaný uživatel přidán do databáze služby Auth0, což v případě existence serverové části aplikace, která má vlastní databázi uživatelů, způsobí problém v podobě nekonzistence registrovaných uživatelů. Auth0 se tento problém nově snaží řešit pomocí tzv. **Hooks**[13], ve kterých lze, pomocí Node.js kódu, definovat akci, která se má vykonat v případě výskytu události, se kterou je hook spjat. Akce může představovat **HTTP POST** na vlastní server, který způsobí přidání uživatele i databáze aplikace. Mezi hooky patří například **Post User Registration Hook**, který je spuštěn po každé registraci nového uživatele s informacemi o tomto uživateli. Problém je, že hook zatím podporuje pouze běžnou registraci pomocí emailu a hesla. Registruje-li se uživatel pomocí třetí strany, hook není vykonán. Naštěstí poskytuje služba i možnost tzv. **Rules**[14], které jsou vykonány pokaždé, když se uživatel autentizuje vůči Auth0. Jelikož lze k uživateli ukládat kromě jeho dat i tzv. **metadata**, je možné napsat rule, které zjistí, zda jde o první pokus o autentizaci uživatele za pomoci metadat, v takovém případě pošle **HTTP POST** na server aplikace a uloží do metadat informaci o vytvoření nového uživatele v databázi aplikace. Při další autentizaci se rule znovu neuplatní.

Princip přihlašování

1. Uživatel zadá přihlašovací údaje, které se odešlou službě Auth0. / Vyžádá přihlášení pomocí třetí strany.
2. Auth0 ověří zaslané přihlašovací údaje. / Přesměruje uživatele na přihlašovací službu třetí strany.
3. Auth0 vrací aplikaci odpověd v podobě **JWT** tokenu / chybové zprávy.
4. Aplikace využije získaný token pro získání informací o uživateli z databáze uživatelů aplikace již na svém serveru.
5. Server odpovídá informacemi o uživateli a uživatel je úspěšně přihlášen.

4.4 JWT

JWT nebo-li **JSON-Web Token** je otevřený protokol, pro bezpečné přenášení informací mezi dvěma stranami v podobně **JSON** objektu [6]. Přenášený objekt je podepsán sdíleným klíčem, případně dvojicí veřejného a soukromého klíče, čímž je možné zaručit důvěrnost. Výhodou JWT tokenu je jeho malá velikost a je jej možné přenášet pomocí URL, HTTP POST nebo pomocí HTTP hlavičky [15]. Díky této vlastnosti se hodí pro autentizaci uživatele vůči serveru v případě bezstavového přenosu pomocí HTTP, kde je zaslán v hlavičce HTTP požadavku[15].

Samotný token se skládá se tří částí, kde jednotlivé části jsou odděleny tečkou a mají podobu JSON objektu. První část tvoří tzv. **header–hlavička** tokenu, která uvádí, že se

[13] https://auth0.com/docs/hooks
[14] https://auth0.com/docs/rules
[15] `Authorization: Bearer <token>`

jedná o token typu JWT a specifikuje, který hashovací algoritmus je používán ve třetí části tokenu. Druhou část tokenu tvoří tzv. **payload—obsah**, který nese informace o uživateli uložené serverem. Některé klíče objektu payloadu jsou vyhrazené podle standartu RFC 7519 a mají svůj význam, např. `exp` představuje unix časové razítko vypršení platnosti tokenu [11]. Do této části tokenu je možné uložit `id` uživatele, podle kterého lze na serveru získat uživatele, který poslal požadavek. JSON objekty první a druhé části jsou převedeny do textového řetězce pomocí **Base64Url**[16] kódování. Řetězce jsou odděleny tečkou. Poslední část tokenu tvoří tzv. **signature—podpis**, který je výsledkem hashovací funkce uvedené v hlavičce tokenu za použití sdíleného klíče. Hashovací funkce dostává jako vstupní hodnotu textový řetězec prvních dvou částí tokenu. Příklad finálního tokenu je možné vidět na obrázku č. 4.2.

Obrázek 4.2: Ukázka JWT tokenu. Jednotlivé části jsou odděleny tečkou.

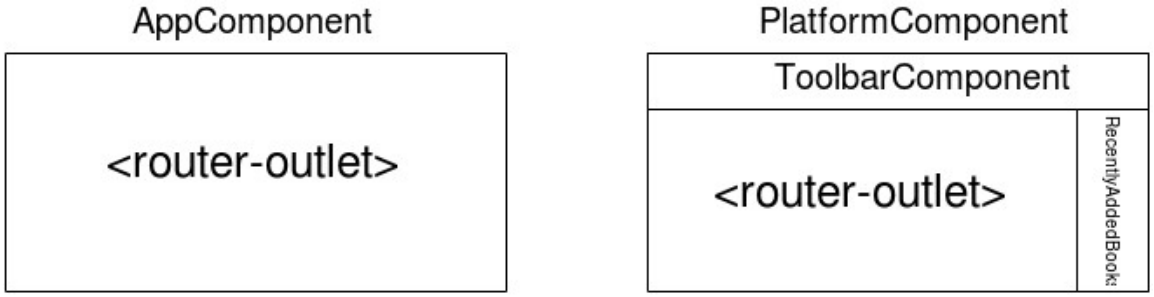
4.5 Klient

Klientská část aplikace je implementovaná pomocí frameworku **Angular 2** viz. kapitola č. 4.1. Kapitola se zmiňuje o tzv. komponentách, které jsou klíčovými prvky aplikace a lze je různě zanořovat. Aplikace je ve výsledku pouze jednou velkou komponentou tzv. `AppComponent`, která dynamicky zobrazuje další podkomponenty pomocí `<router-outlet>` na základě aktuálního stavu routeru.

Aplikace využívá dvou hlavních podkomponent: `LandingComponent` a `PlatformComponent`. `LandingComponent` představuje komponentu, kterou uživatel vidí v případě, že není přihlášen. Lze se zde do aplikace přihlásit nebo registrovat. `PlatformComponent` představuje komponentu, která je dostupná uživateli, jež je úspěšně přihlášen. Tato komponenta má na starost zobrazování komponent jednotlivých částí aplikace, komponenty hlavního panelu a nedávno přidaných knih. Tuto strukturu zobrazuje obrázek č. 4.3.

Kapitola č. 4.1 se také zmiňuje o tzv. servisech, které slouží ke komunikaci se serverovou částí aplikace. Nicméně existuje i další možnost, jak efektivněji komunikovat se serverem a ukládat si vyžádaná data. Klient až na několik málo výjimek nevyužívá servisů, ale využívá tzv. **ngrx/store**[17]. Jedná o stavový kontejner, který je inspirovaný koncepty kontejneru **Redux**[18]. Snaha storu je držet stav celé aplikace v jednotné struktuře, ze které je možné získávat příslušná data. Tím, že jsou data centralizovaná do jedné struktury, namísto roztroušena po jednotlivých servisech, je hledání chyb či zvláštností chování aplikace mnohem snadnější. Store se skládá se čtyř hlavních částí: **Store, Action, Reducer** a **Effect**.

[16] https://en.wikipedia.org/wiki/Base64
[17] https://github.com/ngrx/store
[18] https://github.com/reactjs/redux



Obrázek 4.3: Struktura aplikace. V levé část je zobrazena `AppComponent`, která pouze dynamicky zobrazuje podkomponenty. V pravé části je zobrazena `PlatformComponent`, která zobrazuje komponenty podsekcí této části aplikace.

Store

Store představuje zmíněnou strukturu, ve které jsou uložena stavová data aplikace. Zastupuje tedy celý stav aplikace. Data jsou aplikaci přístupná pomocí tzv. **Observables**[19]. Observable si lze představit jako zdroj dat, ke kterému je možné se přihlásit a v případě, že se data změní, observable deleguje nová data všem částem aplikace, které se k odběru přihlásily.

Struktura storu je koncipovaná jako imutabilní—neměnná, a tak je při každé změně vytvořen zcela nový objekt a starý zahozen, což poskytuje jistotu, že data jsou vždy aktuální. K vyvolání změny stavu aplikace se používají **Actions**.

Actions

Actions představují v ngrx/store možnost vyvolání změny stavu aplikace. Skládají se ze dvou částí: **type** a **payload**. Type představuje o jaký typ action se jedná, např. přidej knihy do poličky uživatele. Payload je volitelná část a představuje data, která si s sebou action nese, např. zmíněná kniha, která má být přidána do poličky uživatele. Akce jsou vyvolávány—dispatchovány podle potřeb aplikace nebo pomocí **Effects**. Nicméně akce jako taková změnu neprovede, pouze ji vyvolá. Fyzickou změnu provede **Reducer**.

Reducer

Reducer představuje funkci, která dostává na vstupu aktuální stav aplikace a action, která byla vyvolána. Na základě vyvolané action provede změnu stavu a nový stav vrátí. Reducer také zastupuje jednotlivé části struktury, např. existuje-li ve struktuře store klíč `shelves`, musí k němu existovat odpovídající reducer, aby bylo možné editovat tuto část store.

Effect

Effect slouží k asynchronnímu získávání dat ze serverové části. Na action, kterou vyvolá aplikace nemusí reagovat jen reducer, ale i effect. Lze tedy vyvolat action, která požaduje např. získání poliček uživatele, na kterou reaguje pouze effect, a to tím, že pošle HTTP požadavek na server. Ve chvíli, kdy dorazí odpověď od serverové části, effect na základě úspěchu nebo neúspěchu požadavku vyvolá příslušnou action, která v payload nese data

[19] http://reactivex.io/documentation/observable.html

poskytnutá serverem. Na tuto akci reaguje příslušný reducer a adekvátně vytvoří nový stav aplikace. Změny jsou následně delegovány všem přihlášeným částem aplikace.

4.5.1 Princip přihlašování

Logika přihlašování do aplikace je umístěna v `AppComponent`, která je hlavní komponentou aplikace a rozhoduje také o tom, která podkomponenta obsahu stránky má být zobrazena. Rozhodování závisí od toho, zda je uživatel přihlášen či nikoli. Po vytvoření komponenty se komponenta přihlásí k odběru dat přihlášeného uživatele ze `store` a následně se dotáže `AuthService`, zda je v `localeStorage`[20] platný JWT token.

Pokud token není platný komponenta přikáže `routeru` změnit `url` stránky na adresu odpovídající `LandingComponent`, která se stará o autentizaci. `LandingComponent` obsahuje dva formuláře, přičemž jeden je určený pro přihlašování a druhý pro registraci. Po vyplnění přihlašovacích, případně registračních údajů komponenta využije funkcí servisu `AuthService`, který pomocí Angular knihovny, poskytnuté službou Auth0[21], zajistí registraci nebo přihlášení uživatele. Proběhne-li registrace úspěšně, získá aplikace platný JWT token, který se uloží do `locaStorage` a následně se celé okno aplikace aktualizuje, čímž dojde k znovuvytvoření `AppComponent` a opět dojde k ověření tokenu, který je již platný. Obdobně funguje i přihlášení pomocí sociálních sítí, ovšem uživatel je přesměrován na stránky těchto sítí a je následně navrácen do aplikace, kde se uloží platný token.

V případě platného tokenu vyvolá komponenta akci pro získání informací o přihlášeném uživateli, na kterou reaguje `ffect` a pošle `HTTP GET` na serverovou část, kde v hlavičce požadavku je použit získaný `token`, čímž může server spojit uživatele s požadavkem a vrátit tak jeho informace. Na základě odpovědi serveru je vyvolána další akce, na kterou reaguje `reducer` a ukládá do stavu aplikace informace o uživateli. Jelikož se stav aplikace změnil a změnila se sekce uživatele, dostává `AppComponent` nové informace o uživateli, což je pro ni znamení, že může zobrazit `PlatformComponent`, čímž se uživatele dostává k obsahu aplikace.

Projití všech těchto kroků se vykoná pouze v případě vytváření `AppComponent`, která se vytvoří pouze, je-li celá stránka aktualizovaná např. pomocí klávesy `F5`. Jinak je tato komponenta stále vytvořená a pouze se mění obsah její podkomponenty `PlatformComponent` za pomoci `routeru`.

4.5.2 Princip překladu textů

Za výběr a použití jazyka je zodpovědný service `LanguageService`, který využívá knihovny **ng2-translate**[22]. Tato knihovna využívá tzv. **Pipe**[23], což je Angular 2 prvek, který se dá považovat za funkci, jež zformátuje data na svém vstupu před jejich zobrazením. Tuto funkci Angular 2 volá automaticky je-li použita v template komponenty[24]. Pipe této knihovny dostává na vstup cestu v tečkové notaci JSON objektu. Ke každému jazyku existuje vlastní `.json` soubor, v nemž jsou v odpovídajícím zanoření překlady textů zobrazené po vykonání pipe.

[20] Úložný prostor pro každou stránku, který poskytuje prohlížeč.
[21] https://auth0.com/docs/quickstart/spa/angular2/00-intro
[22] https://www.npmjs.com/package/ng2-translate
[23] https://angular.io/docs/ts/latest/guide/pipes.html
[24] Příklad užití Pipe: `<div> {{text | customPipe}} </div>`

4.5.3 Specializované komponenty

V této části budou popsány některé komponenty, které aplikace využívá. Komponenty lze rozdělit do dvou kategorií, podle jejich zodpovědnosti. Jedná se o komponenty **smart—chytré** a **dumb—hloupé**. Hloupá komponenta je taková komponenta, která nijak neovlivňuje stav aplikace a nekomunikuje se serverovou částí. Jejím účelem je zpravidla jen zobrazovat data, která jsou ji předána chytrou komponentou pomocí tzv. **@Input**[25] a v případě, že uživateli nabízí možnost interakce, tak pouze vyvolat událost pomocí **@Output**[26], na kterou reaguje chytrá komponenta, ve které se hloupá komponenta nachází. Příkladem hloupé komponenty je například `BookComponent`, která se vyskytuje hned v několika chytrých komponentách. Po kliknutí na tuto komponentu dojde k přesměrování uživatele do detailní sekce této knihy, ovšem komponenta knihy pouze vyvolá událost o tom, že na ni uživatel klikl. Až nadřazená chytrá komponenta reaguje na tuto událost tím, že pomocí `routeru` přesměruje uživatele. Nicméně se v některých případech může stát, že chytrá komponenta obsahuje kromě hloupých komponent i komponenty chytré. Jedním z případů je například `PlatformComponent`, jež je komponentou chytrou, ale obsahuje například `ToolbarComponent`, která je také chytrou komponentou.

PlatformComponent

Jedná se o chytrou komponentu, jejíž hlavní náplní je zobrazovat obsah jednotlivých sekcí aplikace pomocí `<router-outlet>`. Dále je zde umístěna `ToolbarComponent`, představující hlavní navigační panel aplikace včetně vyhledávacího okna a dalších položek. Komponenta dále pomocí `BookComponent` zobrazuje v pravé části okna naposledy přidané knihy, které získává ze `store` a které co dvě minuty aktualizuje vyvoláním příslušné action, což způsobí získání nových dat ze serveru.

BooksComponent

Jedná se o chytrou komponentu zobrazující se pomocí `<router-outlet>` v komponentě `PlatformComponent`. Jejím účelem je zobrazovat všechny knihy uživatele (`BookComponent`), které získá vyvoláním příslušné `action` při svém vytvoření. Komponenta dále obsahuje textové pole pro vyhledávání mezi zobrazenými knihami. V momentě, kdy uživatel napíše či smaže znak z textového pole, změní se hodnota proměnné `pattern`, se kterou je textové pole propojeno pomocí tzv. `two-way bindingu`[27]. Na změnu reaguje funkce `search()`, která vyfiltruje z uživatelových knih pouze takové, jejichž název či autor obsahuje podřetězec proměnné `pattern`. Komponenta také obsahuje vysouvací nabídku tvořenou komponentou `DropdownComponent`, která umožňuje vyfiltrovat pouze knihy podle vybraného stavu. Po kliknutí na jednu z možností nabídky se zavolá funkce `selectBooks()`, která provede vyfiltrování. Jednotlivé knihy jsou dynamicky zobrazovány pomocí direktivy `*ngFor`. Komponenta disponuje tlačítkem pro vygenerování seznamu knih. Po kliknutí na toto tlačítko se pomocí servisu `PdfService` pošle `HTTP GET` na server, který příslušný seznam vygeneruje a zašle jej nazpět aplikaci.

[25] https://angular.io/docs/ts/latest/cookbook/component-communication.html

[26] https://angular.io/docs/ts/latest/guide/template-syntax.html#!#inputs-outputs

[27] Funkce, kterou poskytuje Angular 2. Kdykoli se změní hodnota proměnné ve `View` nebo `Controlleru` komponenty, promítne se tato změna i do sekce, která změnu nevyvolala. Viz. https://angular.io/docs/ts/latest/guide/template-syntax.html#!#two-way

ShelvesComponent

Jedná se chytrou komponentu, která se zobrazuje uvnitř komponenty `PlatformComponent` pomocí `<router-outlet>`. Lze zde najít poličky uživatele, které komponenta získává při svém vytváření pomocí vyvolání příslušné `action`. Komponenta poskytuje možnost přidání poličky pomocí textového okna. Textové okno je opět pomocí two-way bindingu propojené s proměnnou komponenty. Při kliknutí na tlačítko pro přidání poličky se vyvolá `action` jejíž `payloadem` je název nové poličky. Na `action` reaguje `effect` zasláním `HTTP POST` na server, který v případě úspěchu vrátí nově vytvořenou poličku, která se přidá jako `payload` `action`, kterou `effect` vyvolá a na kterou reaguje `reducer`, jenž ji přidá do `store`. Jelikož je komponenta přihlášena k odběru dat o poličkách, je jí nový stav poliček zaslán a direktiva `*ngFor` zajistí vykreslení nového seznamu poliček. Jednotlivé poličky tvoří komponenty `ShelfComponent`, kterým jsou dodávány informace o jednotlivých poličkách a na jejichž události `ShelvesComponent` adekvátně reaguje. Např. při vyvolání události `removeShelf` vyvolá `ShelvesComponent action`, která má odstranit poličku a jako `payload` předá `id` poličky, jež `ShelfComponent` zaslala při vytvoření události. Na `action` reaguje `effect`.

ShelfComponent

Jedná se o hloupou komponentu, která dostává data o poličce, které následně zobrazí. Komponenta používá další dvě hloupé komponenty. `DropdownRowComponent`, což je komponenta sloužící jako vysouvací nabídka s dvěma částmi. Hlavičkový řádek, který se po kliknutí zvětší a zobrazí druhou skrytou část s obsahem a po opětovném kliknutí ji opět skryje. Tato komponenta je použita z důvodu úspory místa, jelikož polička může obsahovat mnoho knih. Na hlavičkovém řádku je možné nalézt textové okno, kterým lze editovat název poličky. Existuje také možnost poličku smazat případně kopírovat. Všechny tyto možnosti vyvolají událost, na kterou reaguje až nadřazená chytrá komponenta. Druhou hloupou komponentou je `BookComponent`, která představuje jednotlivé knihy poličky, které se generují pomocí direktivy `*ngFor`. Kliknutím na tuto komponentu dojde opět k vyvolání události, která vyústí v přesměrování uživatele do detailní sekce této knihy, ovšem až na žádost chytré komponenty, jíž může být `ShelvesComponent` nebo `FriendDetailComponent`.

StatisticsComponent

Jedná se o chytrou komponentu mající na starost zobrazit přehled čtení uživatele a zobrazuje se uvnitř komponenty `PlatformComponent`. Při svém vytváření vyvolává několik `actions`, pro získání potřebných dat ze serveru a také se přihlašuje k odběru těchto dat ze `store`. Data se týkají celkového přehledu čtení uživatele, které je možné vidět v první části komponenty a dále všech čtení knih uživatele bez jejich intervalů, ježje možné vidět v poslední části komponenty zobrazené na časové ose. Poslední data se týkají intervalů čtení knih za aktuální měsíc a je možné je vidět ve druhé části komponenty.

První časová osa nabízí možnost výběru měsíce a roku pro zobrazení intervalů čtení. Po kliknutí na jedno z tlačítek se objeví vysouvací nabídka, která je tvořena komponentou `DropdownComponent`. Když klikneme na jednu z nabízených možností, dojde k zavolání adekvátní funkce, která způsobí vyvolání `action`, která žádá získání nových dat intervalů čtení ze serveru. Jako `payload` této `action` je předán zvolený měsíc a rok. Poté, co jsou nová data doručena komponentě, dojde k jejich transformaci na požadovanou strukturu

a zobrazena na časové ose. Obě časové osy jsou vytvářeny pomocí knihovny **Vis.js**[28] a používají `DropdownRowComponent` z důvodu úspory místa.

ProfileComponent

Jedná se o chytrou komponentu, která se zobrazuje uvnitř komponenty `PlatformComponent`. Komponenta obdobně jako předešlé komponenty získává při vytváření potřebná data a přihlašuje se k odběru dat ze `store`. Data o uživateli jsou zobrazena v levé části komponenty a je možné je editovat pomocí připravených textových polí. Změna je provedena po kliknutí na tlačítko **Potvrdit změnu** vyvoláním příslušné `action`. Dále je možné nahrát nový avatar pomocí `<input>` typu `file`, který se objeví po kliknutí na aktuální avatar. Po vybrání nového avataru dojde k vyvolání příslušné `action`, což způsobí nahrání avataru na server a aktualizaci avataru. Není-li uživatel přihlášen pomocí sociální sítě, je zde nabídnuta možnost změny hesla, kterou zajišťuje služba **Auth0** po zavolání příslušně funkce servisu `AuthService`.

V pravé části komponenty se nachází textové okno, kde lze vyhledávat uživatele aplikace. Při psaní do tohoto okna se zasílají `HTTP` požadavky s textovým řetězcem uvedeným v textovém okně. Požadavek je odeslán v případě, že je hodnota různá od hodnoty před půl vteřinou. Vyhledané výsledky jsou díky `store` doručeny komponentě a ta je pomocí `*ngFor` vykreslí uživateli. Má-li uživatel s vyhledaným uživatelem již nějaký vztah, jsou pomocí direktiv `*ngIf` zobrazena tlačítka odpovídající tomuto stavu. Kliknutím na jedno z tlačítek dojde k vyvolání `adekvátní action`, čímž dojde opět k odeslání `HTTP` požadavku na základě významu tlačítka.

Žádosti o přátelství jsou zobrazeny komponentami `FriendRequestComponent`, které jsou hloupé a zobrazují informace o žádosti. Pokud se jedná o příchozí žádost zobrazuje komponenta pomocí `*ngIf` tlačítka na přijmutí nebo odmítnutí žádost, v opačném případě tlačítko pro zrušení žádosti. Kliknutím na tlačítko dojde k vyvolání události, na kterou reaguje chytrá `ProfileComponent`, která vyvolá odpovídající `action`. Komponenty jsou generovány pomocí `*ngFor` na základě dvou polí žádostí, které vyfiltruje `ProfileComponent` při obdržení dat ze `store`.

Poslední část tvoří přátelé uživatele, kteří jsou vygenerování pomocí `*ngFor`. Jedná se o hloupé komponenty `FriendComponent`. Po kliknutí na ně dojde k přesměrování na detail přítele pomocí `routeru`.

FriendDetailComponent

Jedná se o chytrou komponentu, která se opět zobrazuje uvnitř `PlatformComponent`. Na starost má zobrazení informací o vybraném příteli uživatele. Při vytváření komponenty, se z `url` adresy získá `id` přítele. Toto `id` se použije při vyvolání `action`, která žádá získání informací o tomto uživateli. Komponenta se následně přihlásí k odběru těchto informací ze `store`. Po přijetí uživatelských dat komponentou jsou data vykreslena. Příchozí data obsahují informace o uživateli, které jsou zobrazeny pomocí komponenty `FriendComponent`. Data dále obsahují informace o jednotlivých poličkách, které jsou zobrazeny direktivou `*ngFor` pomocí komponent `ShelfComponent`. Poslední části komponenty jsou samotné knihy uživatele, které jsou zobrazeny komponentami `BookComponent` opět pomocí direktivy `*ngFor`. `FriendDetailComponent` reaguje na události svých hloupých podkomponent obdobně jako předešlé chytré komponenty.

[28] http://visjs.org/index.html

BookDetailComponent

Jedná se o chytrou komponentu, která tvoří stěžejní část aplikace. Opět je zobrazena uvnitř komponenty `PlatfromComponent` a jedná se o největší komponentu aplikace. Při vytvoření komponenty se komponenta přihlásí k odběru všech potřebných dat ze `store`. Získá `id` knihy, kterou má zobrazit z `url` a na základě tohoto `id` vyvolá několik `action`, které způsobí získání informací ze serveru spojených s knihou. Získají se informace o samotné knize, které představují obálku knihy, autora nebo doporučené knihy. Další informace představují vztah uživatele ke knize, které specifikují, zda má uživatel knihu mezi svými knihami a na základě těchto informací se pomocí `*ngIf` zobrazí tlačítka spojena s přidáním knihy do knihovny uživatele, případně tlačítka spojená s úpravami stavu knihy, jako je například možnost změnit stav knihy či začít knihu stopovat. Je-li kniha mezi knihami uživatele objeví se také možnost přidat nebo upravit literární rozbor k této knize, který je řešen pomocí hloupé komponenty `EducationalComponent`, která se dynamicky zobrazuje a skrývá pomocí `*ngIf`. Komponenta také získává data týkající se komentářů knihy. Komentáře jsou zobrazeny pomocí hloupé komponenty `CommmentComponent`, která se podle vstupního přepínače chová staticky—zobrazené komentáře, a nebo dynamicky—komentář uživatele, který je možné upravovat. Z komentářů je při přijetí vyfiltrován komentář uživatele, který je zobrazen v levé části a zbylé komentáře jsou zobrazeny v pravé části pomocí `*ngFor`.

Komponenta je také přihlášena k odběru informací o uživatelském čtení této knihy. Ve chvíli obdržení těchto informací projde veškerá čtení včetně intervalů těchto čtení. Jednotlivé intervaly obsahují časová razítka začátku a konce čtení, která jsou převedena na dobu trvání v milisekundách a následně zformátována pomocí dostupných funkcí do lidštější podoby. Po skončení tohoto procesu je spočteno, jak dlouho uživatel knihu četl a jak dlouhé byly jednotlivé intervaly. Vše se uloží do proměnné, ze které se poté dynamicky zobrazují jednotlivá čtení v nabídnuté tabulce pomocí `*ngFor`. Komponenta je také schopna poznat, zda je aktuálně čtená kniha knihou, která je právě detailně zobrazena a na základě toho dynamicky upravovat zobrazená tlačítka pro zahájení či ukončení stopování knihy. Využívá zde přihlášení k odběru dat ze `store`, tykajících se aktuálně stopované knihy, kde sleduje `id` stopované knihy a `id` zobrazené knihy.

ToolbarComponent

Jedná se opět o chytrou komponentu, která je zobrazena po celou dobu ve vrchní části komponenty `PlatformComponent` a představuje hlavní navigační panel aplikace. Komponenta je přihlášena k odběru informaci o přihlášeném uživateli ze `store`, jehož avatar a uživatelské jméno zobrazuje v pravé části. Kliknutím na na tuto sekci dojde k zobrazení komponenty `ProfileComponent` pomocí routeru. Komponenta je také přihlášena k odběru informací týkajících se žádostí o přátelství. Existuje-li žádost, zobrazí odpovídající ikonu.

Vyhledávání knih je zde zobrazeno pomocí chytré komponenty `SearcherComponent`, která se o vyhledávání stará. Komponenta je přihlášena k odběru informací o vyhledaných knihách, které zobrazuje ve vysouvací nabídce pomocí `*ngFor`. Při změně textu ve vyhledávacím okně se vyvolá `action`, která způsobí odeslání požadavku na server. Díky přihlášení k odběru dat jsou nově vyhledané knihy aktualizovány ve vysouvací nabídce. Kliknutím na jedu z knih dojde k přesměrování na detail této knihy pomocí `routeru`.

Komponenta také obsahuje chytrou komponentu `TrackingBarComponent`, která je přihlášena k odběru dat o posledním stopování knihy uživatele. Její obsah je zobrazen pomocí `*ngIf` v případě, že uživatel již někdy stopoval nějakou knihu. Na základě toho, zda je stopování ukončeno či nikoli, zobrazuje pomocí `*ngIf` adekvátní tlačítko pro spuštění nebo

ukončení stopování, které po kliknutí vyvolá příslušnou `action`. Doba stopování je počítá jako rozdíl aktuálního času a času začátku stopování, který se přepočítává každou vteřinu.

BookComponent

Jedná se o hloupou komponentu, která pouze zobrazuje informace o knize, které jsou ji poskytnuté nadřazenou chytrou komponentou. Komponenta knihy má dva stavy. Je-li v neaktivním stavu, lze vidět obálku knihy přes celou plochu komponenty včetně názvu této knihy. Po najetí kurzoru myši na komponentu dojde pomocí Angular 2 animací ke změně pozice obálky knihy a jejímu zmenšení, čímž se odkryjí další informace o knize, které byly pod obálkou schovány.

4.5.4 Obecné komponenty

Aplikace kromě specializovaných komponent používá i některé obecné komponenty, které zobrazují obsah podle poskytnutého **template** jinou komponentou. Využívá se zde tzv. **Content Projection**. Vloží-li se mezi párový tag Anguar 2 komponenty nějaký obsah, je při vytváření komponenty nahrazen **template** této komponenty a dojde k jeho zahození. Nicméně použije-li **template** komponenty speciální tag `<ng-content></ng-content>`, je obsah mezi párovými tagy komponenty zobrazen na místě použití tohoto tagu, čímž lze rozšířit **template** komponenty nadřazenou komponentou. Takových to rozšíření—template lze použít několik, jelikož lze specifikovat, které rozšíření se má kde zobrazit pomocí více tagů `<ng-content>` obohacených o selektory [12]. Příkladem je `DropdownRowComponent`, která pracuje se dvěma **templaty**.

CardComponent

Jedná se o jednoduchou komponentu, která je použita skoro ve všech komponentách a představuje nastylovaný kontejner, ve kterém je obsah, vložen pomocí `<ng-content>`, zobrazen. Kartu lze na stránce poznat podle průhledného hnědého pozadí.

DropdownComponent

Jedná se o komponentu, která představuje tlačítko, které po kliknutí zobrazí vysouvací nabídku, jejíž obsah je komponentě předán pomocí `<ng-content>`. Komponenta je použita například při výběru stavu knihy v `BookDetailComponent`. Efekt vysunutí je umožněn pomocí Angular 2 animací[29], které umožňují animovat výšku elementu ze stavu `0px` do stavu `auto`, což klasické CSS animace neumožňují.

DropdownRowComponent

Jedná se o komponentu, která představuje klikatelný hlavičkový řádek, jehož vzhled je poskytnutý pomocí `<ng-content>` se selektorem. Po kliknutí se vysune obsah komponenty, který je opět vložen pomocí `<ng-content>` se selektorem. Animace vysunutí je opět umožněna díky Angular 2 animacím.

[29] https://angular.io/docs/ts/latest/guide/animations.html

4.6 Server

Serverová část aplikace je implementovaná pomocí jazyku **Go** viz. kapitola č. 4.2 a několika balíčků dostupných pro tento jazyk. Jelikož se jedná o serverovou aplikaci, bylo nutné vytvořit **API**[30] a umožnit aplikaci poslouchat na určitém portu, aby s ní mohla komunikovat klientská část aplikace pomocí HTTP požadavků. Go nativně poskytuje možnost implementace API, nicméně implementace touto cestou je značně nepříjemná. Z tohoto důvodu byl použit balíček **Echo**[31], který nabízí některé výhody, které aplikace potřebuje a využívá. Pro manipulací s databází byl využit balíček **GORM**[32], který usnadňuje práci s databázemi a značně odkládá nutnost používat jazyk **SQL**[33], jedná-li se o jednoduché dotazy na databázi. Posledním důležitým balíčkem je **Gofpdf**[34], pomocí kterého jsou generována PDF.

Aplikace je rozdělena do dvou modulů, podle jejich účelu. První modul, zastřešuje veškeré `handler` funkce viz. 4.6.1. Ve druhém modulu se nachází jednotlivé modely použité v aplikaci. Hlavní kód aplikace se nachází v souboru `main.go`, ve kterém se pomocí balíčku GORM vytvoří databáze, následně se vytvoří API pomocí Echo, ke kterému se zaregistrují potřebné `handlers`. Zmíněné balíčky jsou více popsány v následujících podkapitolách včetně vysvětlení některých `handlers` a problémů, které se objevily během implementace.

4.6.1 Echo

Balíček poskytuje jednoduchou implementaci tzv. **routeru**, který je vytvářen registrací dvojic `url : handler`. `Url` představuje tzv. **API endpoint**, což je adresa, na kterou jsou posílány HTTP požadavky z klientské části např. `/books/:id`. `Handler` je funkce, která dostává na vstup kontext HTTP požadavku a jejím výstupem je odpověď pro odesílatele požadavku v mnoha podobách např. **JSON, BLOB**[35] a další. Kromě routeru nabízí balíček i možnost použití `CORS`[36], což je mechanismus, který je nutný mít na serverové části podporovaný kvůli ochranné politice prohlížeče, ve kterém běží klientská část. Prohlížeč by kvůli bezpečnostním důvodům odmítl se serverem komunikovat, jelikož se server nachází na jiné doméně než klientská část, což lze eliminovat použitím CORS, kde se klient se serverem domluví na podmínkách komunikace v požadavku `OPTIONS`, který předchází každému požadavku. Dále balíček umožňuje snadné použití JWT tokenu pro ověření přístupu uživatele k daným zdrojům serveru.

Při spuštění aplikace se pomocí balíčku nastaví použití mechanismu `CORS`, dále se nastaví sdílený klíč pro ověření JWT tokenu a tzv. `skipper` funkce, která definuje, které routy jsou přístupné bez JWT tokenu. Tato funkce se volá před vlastním ověřováním tokenu balíčkem Echo. Poté se zaregistrují všechny `routy` včetně jejich `handlerů` a konkurentní server je spuštěn.

[30] Application Programming Interface. https://cs.wikipedia.org/wiki/API
[31] https://echo.labstack.com/
[32] http://jinzhu.me/gorm/
[33] https://cs.wikipedia.org/wiki/SQL
[34] https://godoc.org/github.com/jung-kurt/gofpdf
[35] Přenáší se pomocí něho například multimédia. https://en.wikipedia.org/wiki/Binary_large_object
[36] https://cs.wikipedia.org/wiki/CORS

4.6.2 GORM

Název balíčku vychází ze dvou pojmů **GO** a **ORM**[37], což je technika, která umožňuje převádění dat relační databáze do modelů objektově orientovaných jazyků, čímž značně usnadňuje práci s databází. Balíček na základě dodaných modelů vytvoří odpovídající entity v relační databázi. Zajištění vytvoření relací v databázi je docíleno pomocí speciálních značek v definici modelu, kde je možné balíčku poskytnout tyto informace. GORM disponuje několika adaptéry k populárním databázím. Aplikace používá systém **SQLite**[38].

Známe-li např. `id` uživatele je možné celý jeho záznam z databáze získat pouhým zavoláním metody `db.First(&user, id)`, kde `db` představuje ukazatel na databázi, `user` ukazatel na proměnnou typu `User`, jež odpovídá modelu, na jehož základě byla vytvořena v tabulce entita a `id` je primární klíč záznamu uživatele. Balíček obsahuje velice podrobné API, které umožňuje skládání složitějších dotazů a chrání před tzv. **SQL injection**[39], což je podsunutí zákeřného SQL kódu, který je vykonán při zpracování SQL dotazu.

4.6.3 Gofpdf

Jedná se o balíček, pomocí kterého aplikace generuje soubory PDF. Balíček poskytuje rozšířené API, díky kterému je možné relativně snadno PDF generovat. Balíček umožňuje výběr formátu papíru, užití vlastních fontů písma, vkládání obrázku atp. Před zapsáním textu do souboru je nutné nejdříve připravit *imaginární pero*, kterému lze nastavit velikost písma, barvu, ohraničení atd. Obdobně se vykreslují geometrické objekty.

4.6.4 Registrace a přihlášení uživatele

Jelikož registraci i přihlašování má na starost služba Auth0 volána z klientské části, řeší tento problém server jen částečně. Při každém požadavku na chráněnou routu se ověří platnost zaslaného tokenu. Je-li token platný je vykonána příslušná `handler` funkce, jinak je klientovi vrácena chyba.

Registrace ve službe Auth0 vyvolá spuštění **Rule**, které způsobí odeslání požadavku na routu `/new` serveru, která je nechráněná. Požadavek v části `body` nese informace o nově registrované uživateli, včetně `authID` uživatele, které se uloží do databáze. Token, který je následně zasílán v požadavcích klienta, obsahuje toto `authID` a na základě něho se získá z databáze uživatel, jenž poslal požadavek z klienta.

4.6.5 Vyhledávání knih

K vyhledávání knih se využívá **Goodreads API**, které tuto možnost nabízí registrovaným uživatelům s několika omezeními. Po přijmutí požadavku od klienta se z kontextu požadavku získá hledaný řetězec, který se použije při dotazu na API Goodreads. Odpověď přichází ve formátu `XML`, který se pomocí metody `Unmarshal` převede do kolekce Go objektů na základě modelu `GoodReadsBook`. Převedená kolekce je následně odeslána jako odpověď klientovi ve formátu JSON.

Před implementací přišlo v úvahu využít k vyhledávání Google Books API, ale z důvodu špatného pokrytí českých titulů bylo nakonec vybráno Goodreads, které má pokrytí lepší. Navíc Google Books posílá jako odpověď obrovské množství informací, které působily až

[37] https://en.wikipedia.org/wiki/Object-relational_mapping
[38] http://sqlite.org/
[39] https://cs.wikipedia.org/wiki/SQL_injection

moc zmatečně. Byla oslovena také služba Databazeknih.cz, kde jsou dostupná veškerá česká vydání titulů, zda je možné využít jejich databázi v rámci této práce, ale žádost byla velice stroze odmítnuta.

4.6.6 Přidávání knih

Při přidávání knih do databáze, zasílá klient na server objekt představující informace o knize, na kterou uživatel klikl. Toto se děje pouze v případě kliknutí na knihu ve vyhledávacím okně nebo v doporučených knihách. V ostatních případech se pracuje s `id` knihy této aplikace. Existují dvě možnosti: kniha je již v databázi a knihu je třeba do databáze přidat. V případě první možnosti se do databáze uloží název, autor, odkaz na obálku knihy a `GoodreadsID` knihy, která se získá ze zaslaného objektu. Dále je třeba najít tutéž knihu na Google Books, což se provede zasláním požadavku na Google Books API. V první verzi implementace vyhledávání na Google Books se vyhledávalo podle **ISBN**, ovšem jelikož každé vydání titulu nese jiné ISBN, byly výsledky velice bídné. Zlepšení přineslo vyhledávání podle názvu knihy, které ovšem není stoprocentní a může se stát, že se najde jiná kniha. Odpověď, kterou zašle Google Books se zpracuje a do databáze se k záznamu knihy přidá i `GoogleBooksID`. Následně se klientské části zašle odpověď, která nese `id` knihy této aplikace se kterým se následně pracuje.

4.6.7 Detail knihy

Vznikne-li požadavek na zaslání detailních informací o knize, obdrží serverová část pouze `id` této knihy. Nejdříve se kniha vyhledá v místní databázi pomocí tohoto `id`. Po vyhledání záznamu je známo i `GoodreadsID` a `GoogleBooksID`. Na API příslušných služeb jsou zaslány požadavky na získání informací o knihách určených získanými `ids`. Odpovědi služeb jsou zpracovány a uloženy do objektu, který je následně zaslán jako odpověď na požadavek klienta. Klient tak získává na jeden požadavek informace o knize ze dvou zdrojů naráz.

4.6.8 Začít stopovat úsek čtení knihy

Jednotlivé úseky—intervaly čtení knihy jsou spojeny s jednotlivými čteními knih. Jedno čtení může obsahovat více intervalů. Vznikne-li požadavek na začátek stopování intervalu knih, musí `handler` funkce nejdříve zjistit, zda uživatel aktuálně nestopuje jinou knihu, jelikož nelze v jeden čas stopovat dvě knihy naráz. V případě, že uživatel aktuálně nějakou knihu stopuje, je nutné ji najít a stopování intervalu ukončit. Následně musí funkce ověřit stav knihy. Knihu lze začít číst z jakéhokoli stavu s tím, že se do stavu **čteno** překlopí automaticky. Je-li kniha již čtena, vytvoří se pouze nový záznam v tabulce `Intervals`, který bude obsahovat časové razítko začátku čtení intervalu. V případě, že jde o nové čtení knihy, musí se kniha nejdříve nastavit do stavu **čteno**. Následně se jí přiřadí nový záznam v tabulce `Readings`, čímž se zapamatuje začátek čtení knihy a následně se vytvoří nový záznam v tabulce `Intervals`, který představuje nový interval tohoto čtení.

4.6.9 Změna stavu knihy

Při změně stavu je nutné zjistit v jakém stavu se kniha nachází před vykonáním změny. Je-li aktuální stav knihy **čteno** a příchozí stav je **přečteno** je nutné ukončit stopování úseku knihy v případě, že je kniha aktuálně stopována a následně ukončit celé čtení této knihy. Potom je možné knihu bezpečně převést do stavu **přečteno**. Je-li aktuální stav knihy

různý od **čteno** a příchozí stav je **přečteno** je nutné vytvořit záznam v tabulce `Readings`, aby uloženo, že uživatel knihu přečetl za nulový čas. Je-li příchozí stav **čteno**, vytvoří se záznam v tabulce `Readings`, který zaznamená začátek čtení knihy. Ostatní stavy pouze přepnou stav knihy bez nutnosti dalších změn.

4.6.10 Generování PDF

Generování PDF probíhá pomoci zmíněného balíčku **Gofpdf**. Na základě typu PDF, podle zaslaného požadavku, se z databáze načtou všechna potřebná data, které jsou poté pomocí metod, nabízených balíčkem, zapsána do nového PDF. Vytvořené PDF se uloží na disk. Z disku je následně načteno zpět do programu a převedeno na pole bytu `[]byte`, aby jej bylo možné odeslat jako **BLOB** v odpovědi uživateli. Soubor je následně z disku odstraněn.

Kapitola 5

Testování

Tato kapitola se zabývá testováním výsledné aplikace. Testování serverové části probíhalo pouze ve formě ověření správné funkčnosti jednotlivých `handler` funkcí po jejich implementaci převážně pomocí nástroje **Postman**[1]. Tento nástroj umožňuje zasílání HTTP požadavků na cílovou adresu včetně pohodlného nastavení sekcí **Body** a **Headers**. Nástroj poskytuje jednoduché uživatelské rozhraní a je vhodný pro rychlé otestování chování serverové části. Nástroje bylo také užito z důvodu předcházející se implementace serverové části před částí klientskou. Po dokončení příslušné sekce na straně klienta byla ověřena funkcionalita i za použití klienta, čímž byly většinou zjištěny některé chyby či chybějící funkcionalita. Přidání možnosti stopovat úsek čtení knihy způsobilo nutnost reimplementovat `handler`, který byl zodpovědný za přepínání stavu knihy, jelikož jeho stávající verze neuměla pracovat s úseky čtení.

Klientská část byla podrobena většímu testování a byla hlavním námětem konzultací z důvodů funkcionality a uspořádání jednotlivých komponent v sekcích. Prvotní navržená kostra celé aplikace byla postupně doplňována a upravována na základě poznatků z konzultací. Poznatky také přinesly nezasvěcené osoby, které aplikaci viděly náhodně během jejího vývoje a vyjádřily k ní své nápady. Jedním z nápadů je zobrazení nedávno přidaných knih v pravé části aplikace. Možnost zkopírovat celou poličku přítele mezi své poličky vznikla také na popud třetí strany.

Samotní uživatelé měli možnosti si aplikace vyzkoušet až po dokončení její implementace. Aplikace byla více než týden vystavena na školním studentském serveru **eva**, kde ji uživatelé mohli volně používat. K poskytnutí zpětné vazby jim byl nabídnut dotazník, který zjišťoval spokojenost uživatelů s používáním aplikace a reakce na služby, které aplikace nabízí oproti konkurenčním službám.

Během vystaví aplikace došlo k odhalení serverové chyby, kterou bylo nutné opravit. Chyba se nacházela v `handler` funkci, která zajišťuje získání detailních informací o knihách. Při zpracovávání odpovědi od Google Books API se neověřovala délka pole autorů knihy, což v jednom případě způsobilo pád, jelikož pole přišlo prázdné, ale funkce i přesto přistoupila k prvnímu prvku tohoto pole.

5.1 Výsledky dotazníku

Vyplnění dotazníku se zúčastnilo celkem jednadvacet osob a bylo jim položeno celkem deset otázek, přičemž dvě byly nepovinné. Nepovinné otázky se týkaly přehlednosti aplikace a

[1] https://www.getpostman.com/

chybějících služeb. Co se týče otázky přehlednosti, lze ji rozdělit na dva tábory. Lidé, kteří se zde vyjádřili, považovali aplikaci buď za absolutně nepřehlednou a nebo přehlednou, ale museli ji nejdříve celou projít, aby se v ní zorientovali. Objevila se i kritika na příliš tmavé barvy a rušivé pozadí aplikace ve spojení s průhledností jednotlivých elementů.

Druhá otázka se týkala chybějící funkcionality aplikace. Za chybějící považovali uživatelé například možnost zobrazit si všechny knihy vyhledaného autora a na základě tohoto výčtu si vybrat knihu, kterou by si rádi přečetli. Což nabádá k vytvoření nové sekce, která by umožňovala pracovat s jednotlivými autory. Další návrh padl na možnost zaznamenat si, na jaké straně úsek čtení knihy započal, a na jaké skončil k vedení si většího přehledu o čtení. Obdobnou možnost nabízí služba Goodreads. Uživatelům také chyběla možnost zpětně zadat dobu celého čtení, případně jednotlivých úseků čtení včetně možnosti tyto data editovat. Uživatelé, jak se zdálo, také očekávali možnost přečíst si výňatek knihy, případně celou knihu v této aplikaci. Tato služba by se dala poskytnout u veřejně dostupných elektronických knih, nicméně tato skupina by tvořila velice malou část a u většiny *bestsellerů* by nebyla stejně dostupná. Nicméně tuto možnost nabízí v určité míře Google Books, na jehož detail knihy se dá dostat přímo z detailu knihy aplikace.

Výsledky v číslech

Tabulka č. 5.1 zobrazuje otázky, které byly nabídnuty k vyplnění v dotazníku včetně počtu kladných a záporných odpovědí. Kromě těchto otázek obsahoval dotazník ještě celkové ohodnocení aplikace na stupnici 1–5, kde jednička představovala nejlepší hodnocení a pětka hodnocení nejhorší. Jednička získala celkem osm hlasů, dvojka hlasů devět a pětka získala hlasy čtyři. Hodnocení tři a čtyři nezískali žádné hlasy.

Otázka	Ano	Ne
Využili jste k registraci některé z nabízených sociální?	14	7
Přišla Vám aplikace přehledná?	16	5
Ocenili byste možnost úvodního tutoriálu, který by pomohl zorientovat se v sekcích aplikace	10	11
Využívali byste možnost přidání literárního rozboru ke knize?	11	10
Využívali byste možnost stopování si doby čtení knihy?	12	9
Používali byste aplikaci pro vedení si přehledu o čtených knihách?	15	6
Uvítali byste mobilní verzi aplikace?	15	6

Tabulka 5.1: Tabulka kladených otázek v dotazníku včetně odpovědí.

Shrnutí

Z výsledků dotazníku, lze určit, že o přihlášení do aplikace pomocí třetích stran projevili zájem více, jak dvě třetiny uživatelů. Pouze necelé třetině přišla aplikace nepřehledná. Přibližně polovina uživatelů by ocenila úvodního průvodce pro orientaci v aplikaci, využila by možnosti vyplnění literárního rozboru či stopování si doby čtení knihy. Dvě třetiny uživatelů by aplikaci používali pro zajištění přehledu o čtených knihách a uvítali by možnost mobilní verze. Dotazník také poukázal na služby, které uživatelům v aplikaci chybí a které by bylo vhodné doimplementovat. Poukázáno bylo i na malou responzibilitu aplikace.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a implementovat aplikaci, která umožní uživateli vést si elektronický čtenářský deník. Implementaci práce předcházel průzkum současných služeb, které se touto tématikou zabývají. Mezi zkoumanými službami byly Goodreads, Google Books a Databazeknih.cz. Byly prozkoumány nabízené možnosti těchto služeb, ze kterých se částečně vycházelo v průběhu implementace. Oproti existujícím službám byla aplikace obohacena možností přidat ke knize vlastní literární rozbor a také možností stopovat dobu čtení jednotlivých knih, což bylo hlavním důvodem vzniku této práce.

Aplikace využívá architektury klient-server a tak vznikly aplikace celkem dvě. Klientská část byla implementovaná jako webová stránka pomocí frameworku Angular 2 v podobě jedné dynamicky se měnící stránky. Serverová aplikace byla implementovaná pomocí jazyka Go. Aplikace hojně užívá služby Auth0, která je zodpovědná za registraci uživatelů, jekoliž umožňuje pohodlné užívání uživatelských účtů třetích stran pro autentizaci. Jako hlavní zdroj informací o knihách je použita služba Goodreads, pomocí které jsou knihy i vyhledávány. Dalším zdrojem informací je služba Google Books. Před vlastní implementací byla za nejlepší zdroj informací považovaná služba Databazeknih.cz, ovšem užití zdrojů této služby v aplikaci bylo majitelem zamítnuto.

Testování aplikace uživateli přineslo věcné poznatky pro případný budoucí vývoj. Uživatelé by rádi měli možnost připojit ke stopování úseku čtení knihy i číslo aktuální stránky. Neexistující možnost zpětně editovat či manuálně přidávat úseky doby čtení knih uživateům při testování taktéž chyběla. Také vyvstal požadavek na vytvoření sekce autorů, kde by se dalo zobrazit více informací o autorovi včetně všech jeho děl, což by přispělo k větší pohodlnosti při objevování nových knih. Ovšem případný budoucí vývoj by se měl zaměřit hlavně na responzivitu celé aplikace, aby ji bylo možné využívat i na malých zařízeních.

Literatura

[1] *Angular 2*. [Online; navštíveno 09.05.2017].
URL https://angular.io/

[2] *Auth0*. [Online; navštíveno 09.05.2017].
URL https://auth0.com/

[3] *Databazeknih.cz*. [Online; navštíveno 09.05.2017].
URL http://www.databazeknih.cz/

[4] *Goodreads*. [Online; navštíveno 09.05.2017].
URL https://www.goodreads.com/

[5] *Google Books*. [Online; navštíveno 09.05.2017].
URL https://books.google.com/

[6] *JWT*. [Online; navštíveno 09.05.2017].
URL https://jwt.io/

[7] *The Go Programming Language*. [Online; navštíveno 09.05.2017].
URL https://golang.org/

[8] Angular 2: The architecture diagram. 2017, [Online; navštíveno 09.05.2017].
URL
https://angular.io/resources/images/devguide/architecture/overview2.png

[9] Google: *Google Books Screenshots*. [Online; navštíveno 02.05.2017].
URL https://www.google.com/googlebooks/library/screenshots.html

[10] Jesisem: *Goodreads*. Srpen 2012, [Online; navštíveno 02.05.2017].
URL http:
//nastroje.knihovna.cz/nastroje/citace-a-bookmarking/154-goodreads.html

[11] Jones, M.; Bradley, J.; Sakimura, N.: JSON Web Token (JWT). RFC 7519, RFC Editor, Červenec 2015.
URL https://tools.ietf.org/html/rfc7519

[12] Motto, T.: *Transclusion in Angular 2 with ng-content*. Březen 2016, [Online; navštíveno 10.05.2017].
URL https://toddmotto.com/transclusion-in-angular-2-with-ng-content

[13] Neznámý: *ARCHITECTURE OVERVIEW*. [Online; navštíveno 09.05.2017].
URL https://angular.io/docs/ts/latest/guide/architecture.html

[14] Neznámý: *Documentation*. [Online; navštíveno 09.05.2017].
URL https://golang.org/doc/

[15] Neznámý: *Introduction to JSON Web Tokens*. [Online; navštíveno 09.05.2017].
URL https://jwt.io/introduction/

[16] Neznámý: *Jak napsat čtenářský deník?* [Online; navštíveno 02.05.2017].
URL https://www.seminarkybezprace.cz/blog/jak-napsat-ctenarsky-denik/

[17] Neznámý: *Jaké knihy patří do databáze?* [Online; navštíveno 02.05.2017].
URL http://www.databazeknih.cz/napoveda/jake-knihy-patri-do-databaze

[18] Neznámý: *O projektu DatabazeKnih.cz*. [Online; navštíveno 02.05.2017].
URL http://www.databazeknih.cz/napoveda/o-databazi-knih

[19] The Editors of Encyclopædia Britannica: *Client-server architecture*. [Online; navštíveno 03.05.2017].
URL https://www.britannica.com/technology/client-server-architecture

[20] Vega, J.: *Client-side vs. server-side rendering: why it's not all black and white*. Únor 2017, [Online; navštíveno 09.05.2017].
URL
https://medium.freecodecamp.com/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d

[21] Čermák, M.: *Vícevrstvá architektura: tenký, tlustý a chytrý klient*. Únor 2012, [Online; navštíveno 03.05.2017].
URL http://www.cleverandsmart.cz/vicevrstva-architektura-tenky-tlusty-a-chytry-klient/

Přílohy

Seznam příloh

Příloha A

Obsah přiloženého paměťového média

Přiložené DVD k této práci obsahuje:

- Zdrojový kód klientské části aplikace včetně dokumentace
- Zdrojový kód serverové části aplikace
- Kód Rule služby Auth0
- Zdrojový kód textové části bakalářské práce pro LaTeX
- Text bakalářské práce ve formátu PDF
- Soubor readme.txt

Příloha B

Kód Rule ve službě Auth0

```
function(user, context, callback){
  var userInfo = {};
  // if not in application database yet
  if(!user.app_metadata){
    // if registration was via FB or Google+
    if(user.identities[0].isSocial){
      userInfo.id = user.user_id;
      userInfo.username = user.name;
      userInfo.firstName = user.given_name;
      userInfo.lastName = user.family_name;
      userInfo.imageUrl = user.picture;
    } else {
      userInfo.id = user.user_id;
      userInfo.email = user.email;
      userInfo.username = user.nickname;
    }
    request.post({
      "url": configuration.apiUrl + "/new",
      "json": userInfo
      }, function(err, response, body) {
      if(err || response.statusCode !== 200) {
        return callback(null, user,context);
      }
          // update user's metadata
          auth0.users.updateAppMetadata(user.user_id, {done: true})
          .then(function(){
            return callback(null, user,context);
          });
        }
      );
  }
  callback(null, user, context);
}
```